Benghazi University
Faculty of Information Technology
**Software Engineering Department**

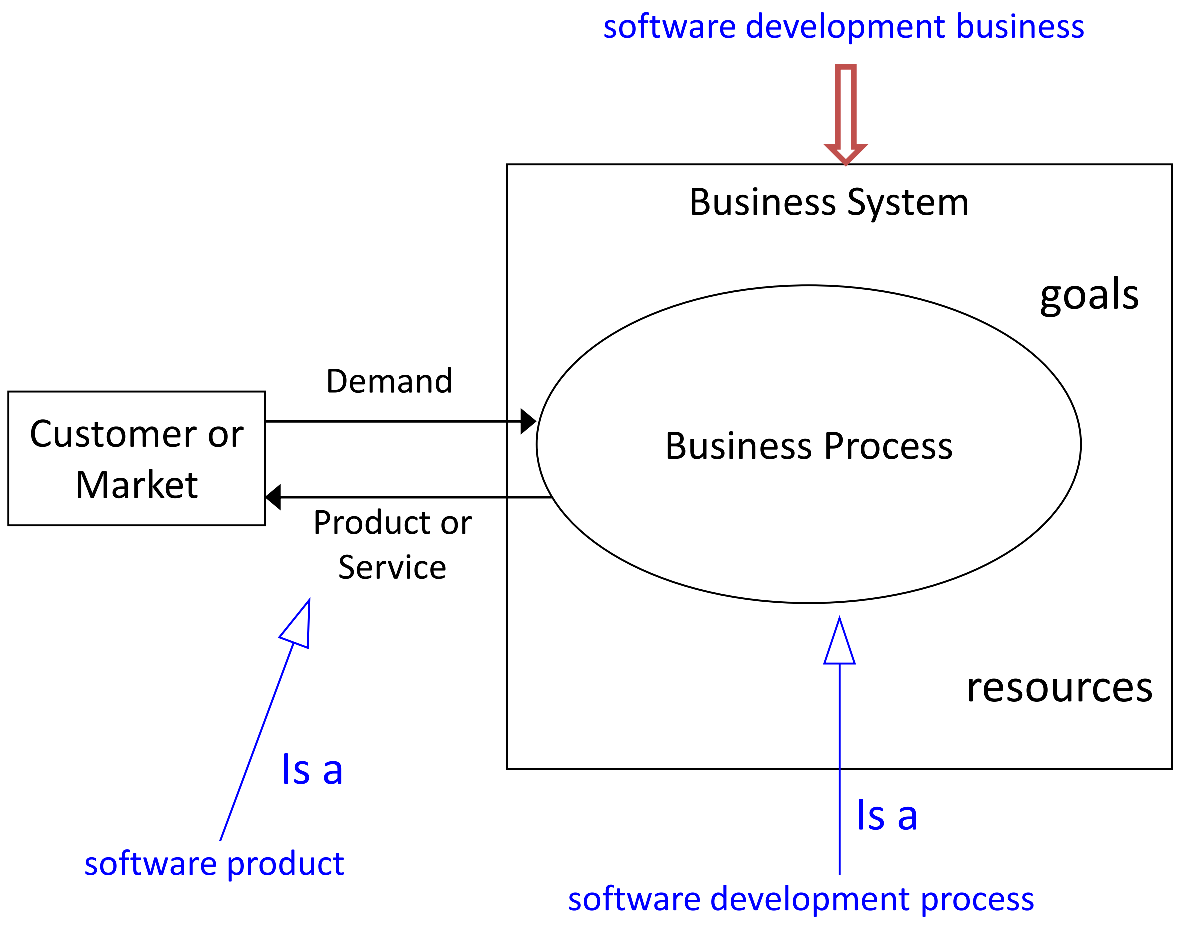
# *Software Quality*

## *Part1*
## *(Introduction)*

***Prepared by: Mohamed A. Elorfi***

***Instructor: Salah Abdelsattar***

# Software Product and process



## Software product

What is a software product?

- Software product = computer programs (sources and executables) + associated documentation + data

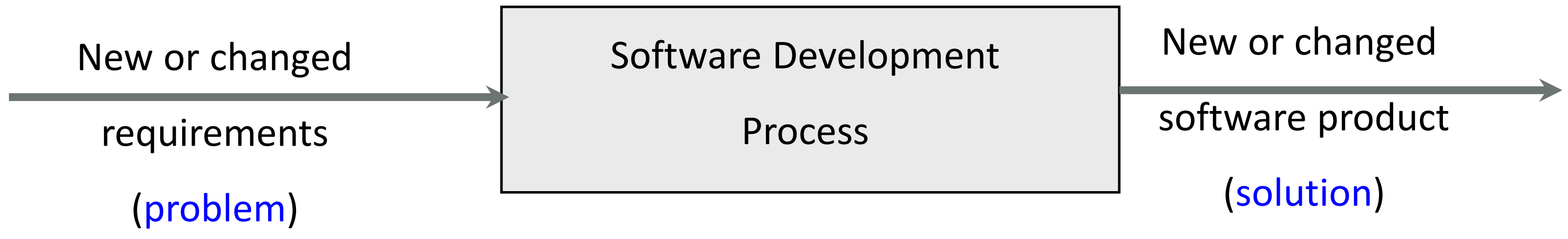
- Software products may be
  - Custom - developed for a particular customer, according to its specifications

  - Generic (“package”) - developed for a general market, to be sold to a range of different customers

## Software process

What is a software development process?

- A set of activities whose goal is the development or evolution of a software product
  - To be followed/instantiated in individual software development projects
- It's the main business process in a software development business
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands



## Software errors, software faults and software failures

■ **Software errors** are sections of the code that are partially or totally incorrect as a result of a grammatical, logical or other mistake made by a systems analyst, a programmer, or another member of the software development team.

■ **Software faults** are software errors that cause the incorrect functioning of the software during a specific application.

■ **Software failures** software faults become software failures only when they are "activated", that is, when a user tries to apply the specific software section that is faulty. Thus, the root of any software failure is a software error.

## Software defects

- What is a software defect?
  - Definition #1: There is a mismatch between the program and its requirements specification.
    - This definition is fine if a requirements specification exists and is complete and correct (not always true).
  - Definition #2: The program does not do what its end user reasonably expects it to do.
    - This definition always applies, even when there's no specification.

## Defect types

Categories of Defects

- Functional defects
  - The program’s features don’t work as they should
- User Interface defects
  - Usability problems
- Performance defects
  - Too slow, Uses too much memory/disk space/bandwidth/etc.
- Error Handling defects
  - Failure to anticipate and handle possible errors, or deal with them in a reasonable way
- Security defects
  - Attackers can compromise the system and access sensitive data or other resources
- Load defects
  - Can't handle many concurrent users, can't handle large data sets
- Configuration defects
  - Doesn't work on the required hardware/OS configurations
- Documentation defects
  - User manuals or online help isn't clear, complete, well-organized

## Defect causes

**A Sample of Possible Causes for Defects**

1. Faulty requirements definition
2. Client-developer communication failure
3. Deliberate deviations from software requirements
4. Logical design errors
5. Coding errors
6. Non-compliance with documentation and coding instructions
7. Shortcomings of the testing process
8. Procedure errors
9. Documentation errors

**It should be emphasized that all causes of error are human, the work of systems analysts, programmers, software testers, documentation experts, and even clients and their representatives.**

## Explanation of Defect causes

### *(1) Faulty definition of requirements*

The faulty definition of requirements, usually prepared by the client, is one of the main causes of software errors. The most common errors of this type are:

■ Erroneous definition of requirements.

■ Absence of vital requirements.

■ Incomplete definition of requirements. For instance, one of the requirements of a municipality's local tax software system refers to discounts granted to various segments of the population: senior citizens, parents of large families, and so forth. Unfortunately, a discount granted to students was not included in the requirements document.

■ Inclusion of unnecessary requirements, functions that are not expected to be needed in the near future.

### *(2) Client–developer communication failures*

Misunderstandings resulting from defective client–developer communication are additional causes for the errors that prevail in the early stages of the development process:

■ Misunderstanding of the client's instructions as stated in the requirement document.

■ Misunderstanding of the client's requirements changes presented to the developer in written form during the development period.

■ Misunderstanding of the client's requirements changes presented orally to the developer during the development period.

■ Misunderstanding of the client's responses to the design problems presented by the developer.

■ Lack of attention to client messages referring to requirements changes and to client responses to questions raised by the developer on the part of the developer.

## Explanation of Defect causes

### *(3) Deliberate deviations from software requirements*

In several circumstances, developers may deliberately deviate from the documented requirements, actions that often cause software errors. The errors in these cases are byproducts of the changes. The most common situations of deliberate deviation are:

■ The developer reuses software modules taken from an earlier project without sufficient analysis of the changes and adaptations needed to correctly fulfill all the new requirements.

■ Due to time or budget pressures, the developer decides to omit part of the required functions in an attempt to cope with these pressures.

■ Developer-initiated, unapproved improvements to the software, introduced without the client's approval, frequently disregard requirements that seem minor to the developer. Such "minor" changes may, eventually, cause software errors.

### *(4) Logical design errors*

Software errors can enter the system when the professionals who design the system – systems architects, software engineers, analysts, etc. – formulate the software requirements. Typical errors include:

■ Definitions that represent software requirements by means of erroneous algorithms.

■ Erroneous definition of boundary conditions.

■ Omission of definitions concerning reactions to illegal operation of the software system.

## Explanation of Defect causes

### *(5) Coding errors*

A broad range of reasons cause programmers to make coding errors. These include misunderstanding the design documentation, linguistic errors in the programming languages, errors in the application of CASE and other development tools, errors in data selection, and so forth.

### *(6) Non-compliance with documentation and coding instructions*

Almost every development unit has its own documentation and coding standards that define the content, order and format of the documents, and the code created by team members.

■ Team members who need to coordinate their own codes with code modules developed by “non-complying” team members can be expected to encounter more than the usual number of difficulties when trying to understand the software developed by the other team members.

■ Individuals replacing the “non-complying” team member will find it difficult to fully understand his or her work.

■ The design review team will find it more difficult to review a design prepared by a non-complying team.

■ The test team will find it more difficult to test the module; consequently, their efficiency is expected to be lower, leaving more errors undetected.

■ Maintenance teams required to contend with the “bugs” detected by users and to change or add to the existing software will face difficulties when trying to understand the software and its documentation.

## Explanation of Defect causes

***(7) Shortcomings of the testing process***

Shortcomings of the testing process affect the error rate by leaving a greater number of errors undetected or uncorrected. These shortcomings result from the following causes:

■ Incomplete test plans leave untreated portions of the software or the application functions and states of the system.

■ Failures to document and report detected errors and faults.

■ Failure to promptly correct detected software faults as a result of inappropriate indications of the reasons for the fault.

■ Incomplete correction of detected errors due to negligence or time pressures.

***(8) Procedure errors***

Procedures direct the user with respect to the activities required at each step of the process. They are of special importance in complex software systems where the processing is conducted in several steps, each of which may feed a variety of types of data and allow for examination of the intermediate results.

## Explanation of Defect causes

***(9) Documentation errors***

The documentation errors that trouble the development and maintenance teams are errors in the design documents and in the documentation integrated into the body of the software. These errors can cause additional errors in further stages of development and during maintenance. Another type of documentation error, one that affects mainly the users, is an error in the user manuals and in the "help" displays incorporated in the software. Typical errors of this type are:

■ Omission of software functions.

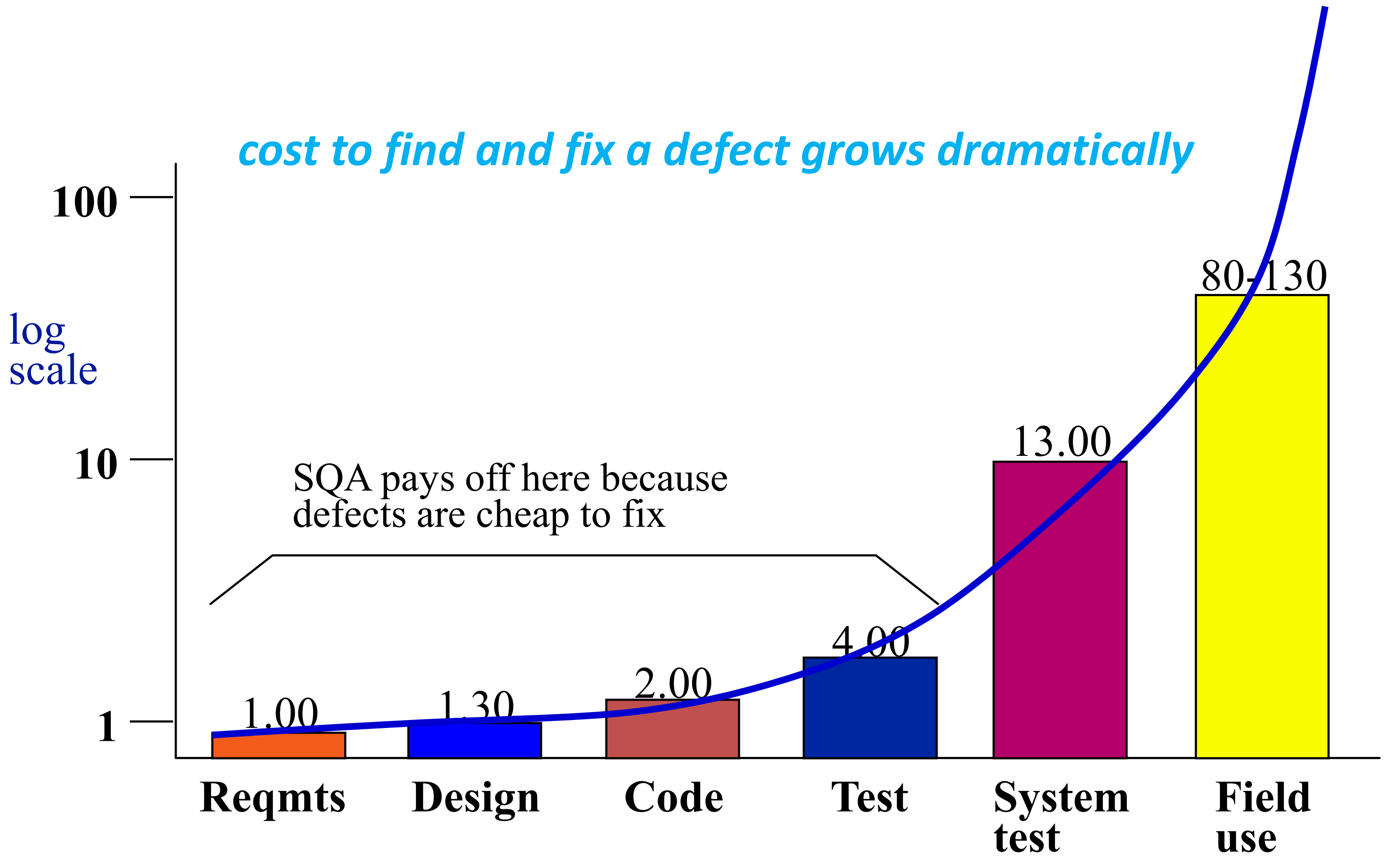
■ Errors in the explanations and instructions given to users, resulting in "dead ends" or incorrect applications.

■ Listing of non-existing software functions, that is, functions planned in the early stages of development but later dropped, and functions that were active in previous versions of the software but cancelled in the current version.

## Cost of defects

- The cost of a defect grows dramatically the longer it remains in the system.
- What is the cost of a defect in the requirements specification if it’s found
  - during requirements phase?
  - during implementation?
  - after product ships?

## Cost of defects


cost to find and fix a defect grows dramatically
log scale
SQA pays off here because defects are cheap to fix
Reqmts
Design
Code
Test
System test
Field use