Unit 4 Systems Engineering and Quality Assurance

Introduction of SEQA

The quality of a system depends on its

- Design
- Development
- Testing
- Implementation

A weakness in any of these areas will seriously risk the quality and therefore the value of the system to the users.

Design Objectives

The two main objectives of design are

Reliability:

- The first is that the system is meeting the right requirements.
- The second level of reliability involves the actual working of the system delivered to the user.

Maintainability:

- When systems are installed, they generally are used for long periods. The average life of a system is 4 – 6 years with the oldest applications often in use for over 10 years.

Program Structure Charts

Well-structured designs improve the maintainability of a system.

The modules should be designed such that they have minimal effect on other modules in the system.

The connections between modules are limited and the interaction of data is minimal.

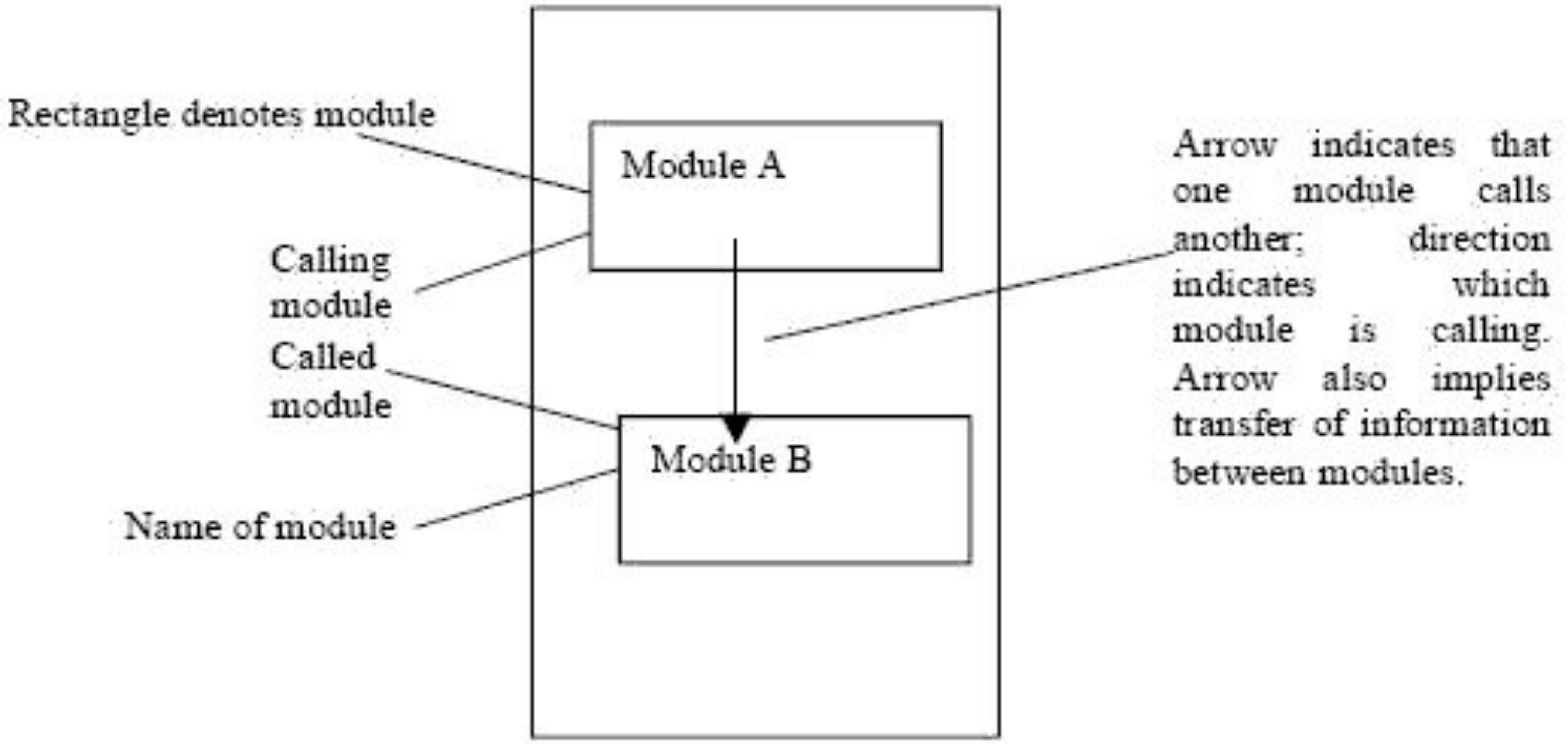
Hence, we need to show the relationships between modules of a system.

The tool used to show this relationship visually is called structure charts.

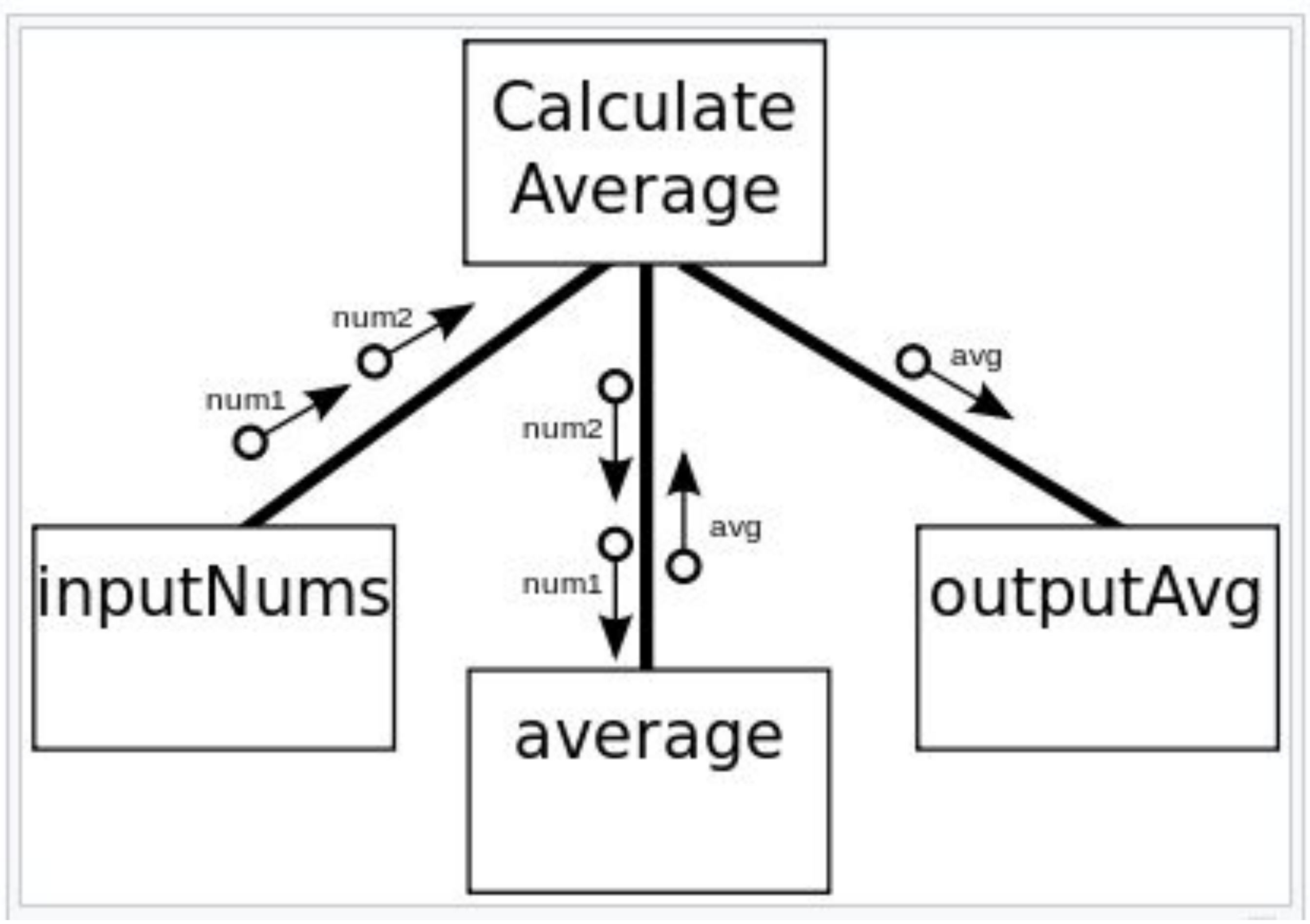
Notation of Structure Charts

A common notation is used in the construction of structure charts for ease of communication among systems developers.

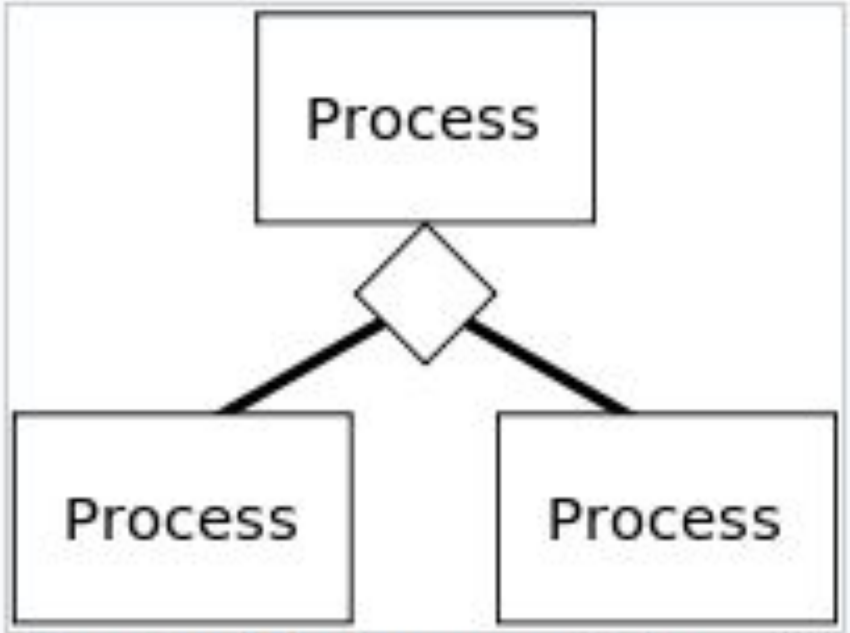
1. Program modules are identified by rectangles with the name written inside the rectangle.
2. Arrows indicate calls between modules.


Rectangle denotes module
Module A
Calling module
Called module
Module B
Name of module
Arrow indicates that one module calls another; direction indicates which module is calling. Arrow also implies transfer of information between modules.

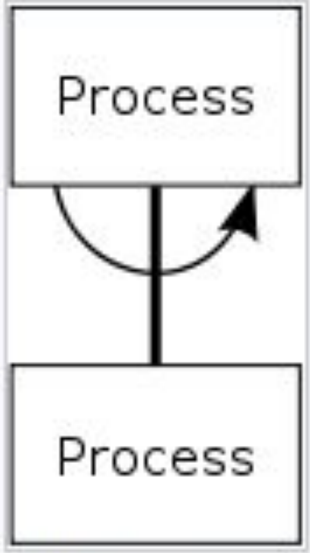
3. Annotations (comments, remarks, notes) on structure charts indicate the parameters that are passed and the direction of data movement.



Selection Notation Iteration Notation



Structure Chart representation of the selection code



Structure Chart of the code to the left

The principles of good software design are as follows

- Modularity and partitioning
- Coupling
- Cohesion
- Span of control
- Size
- Shared use

Modularity and partitioning

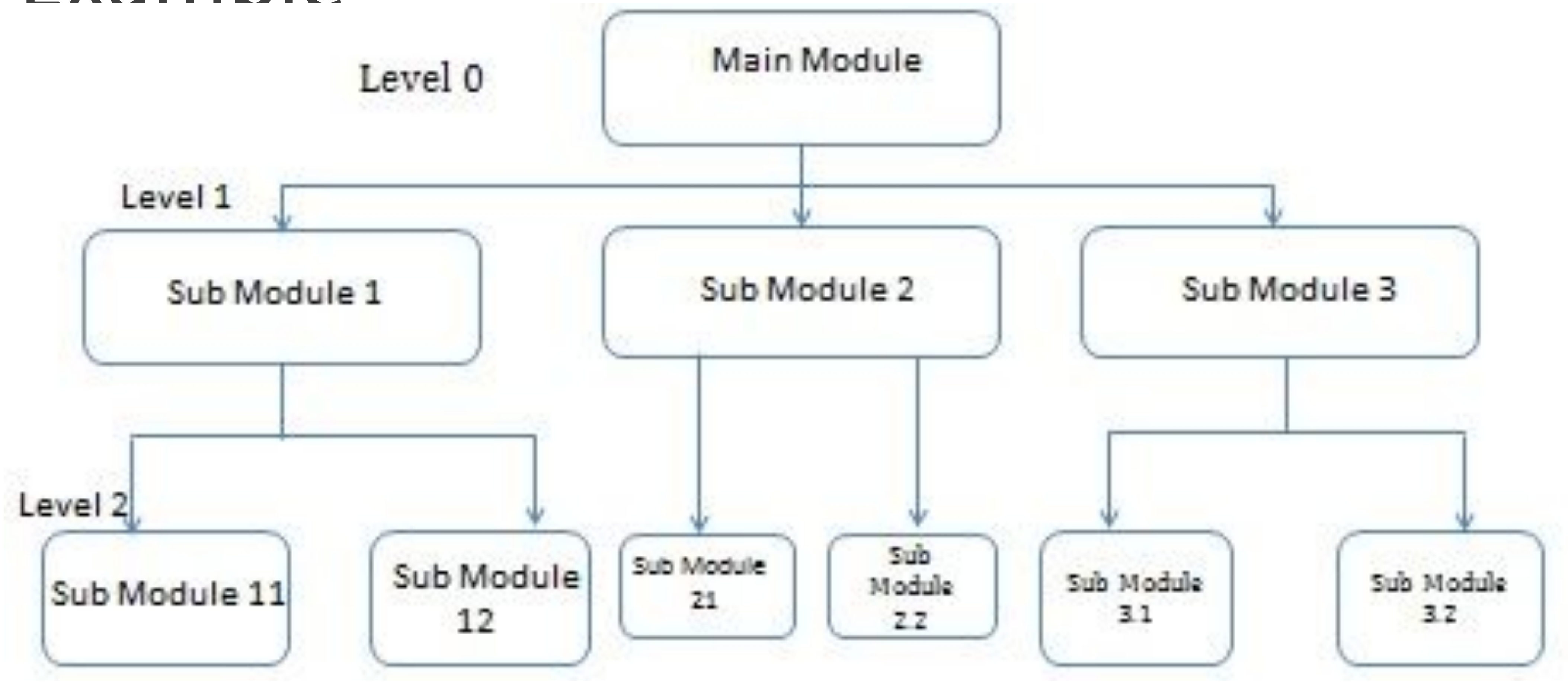
Modularity and partitioning means each system should consist of number of modules and they should be *arranged in a hierarchy*.

Design structure in top-down fashion with modules performing specific function.

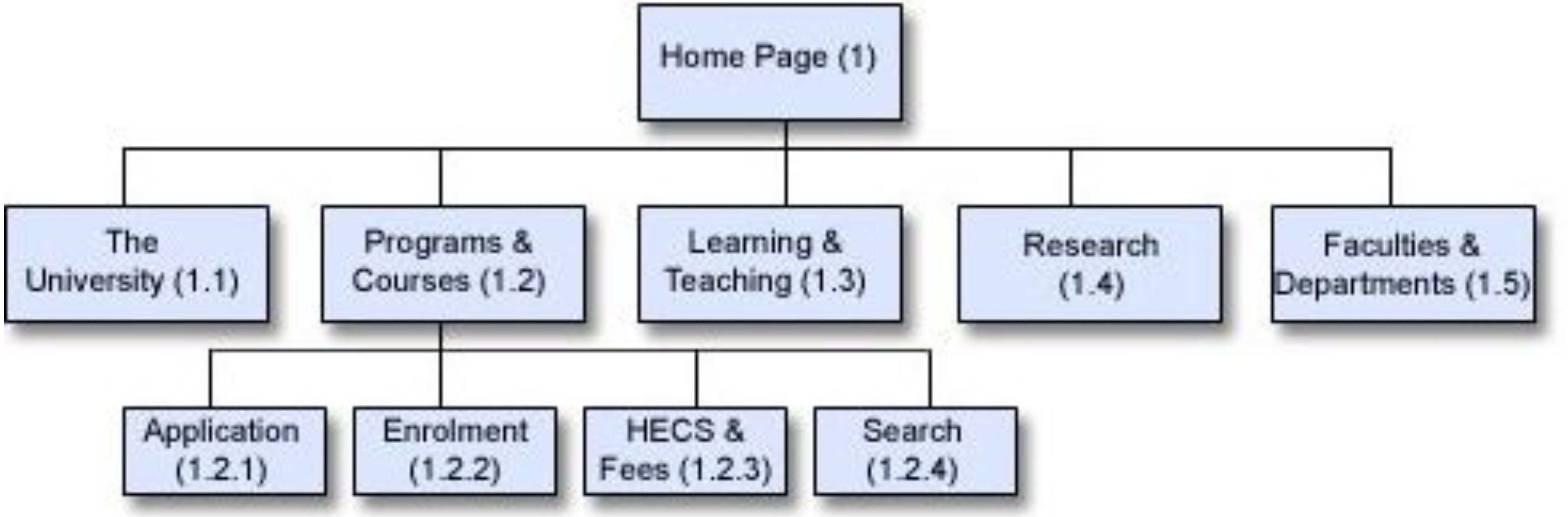
We start at the top with general understanding and gradually move down to levels of greater details.

This is called stepwise enhancement.

Example


Level 0
Main Module
Level 1
Sub Module 1
Sub Module 2
Sub Module 3
Level 2
Sub Module 11
Sub Module 12
Sub Module 21
Sub Module 2.2
Sub Module 3.1
Sub Module 3.2

Example


Home Page (1)
The University (1.1)
Programs & Courses (1.2)
Learning & Teaching (1.3)
Research (1.4)
Faculties & Departments (1.5)
Application (1.2.1)
Enrolment (1.2.2)
HECS & Fees (1.2.3)
Search (1.2.4)

Coupling

Coupling refers to the strength of relationship between modules in a system.

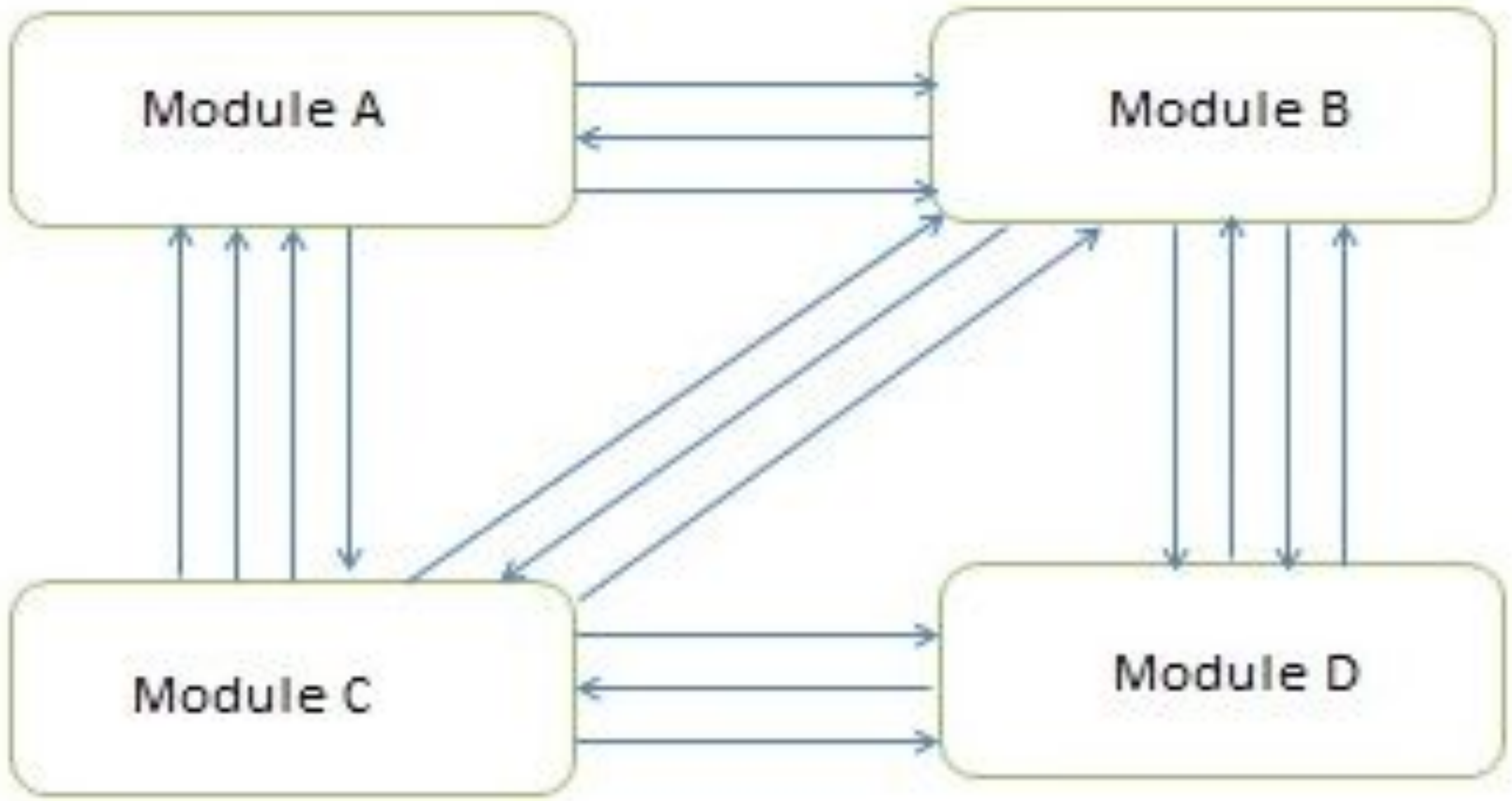
Coupling is a measure of interconnection among modules in a program structure.

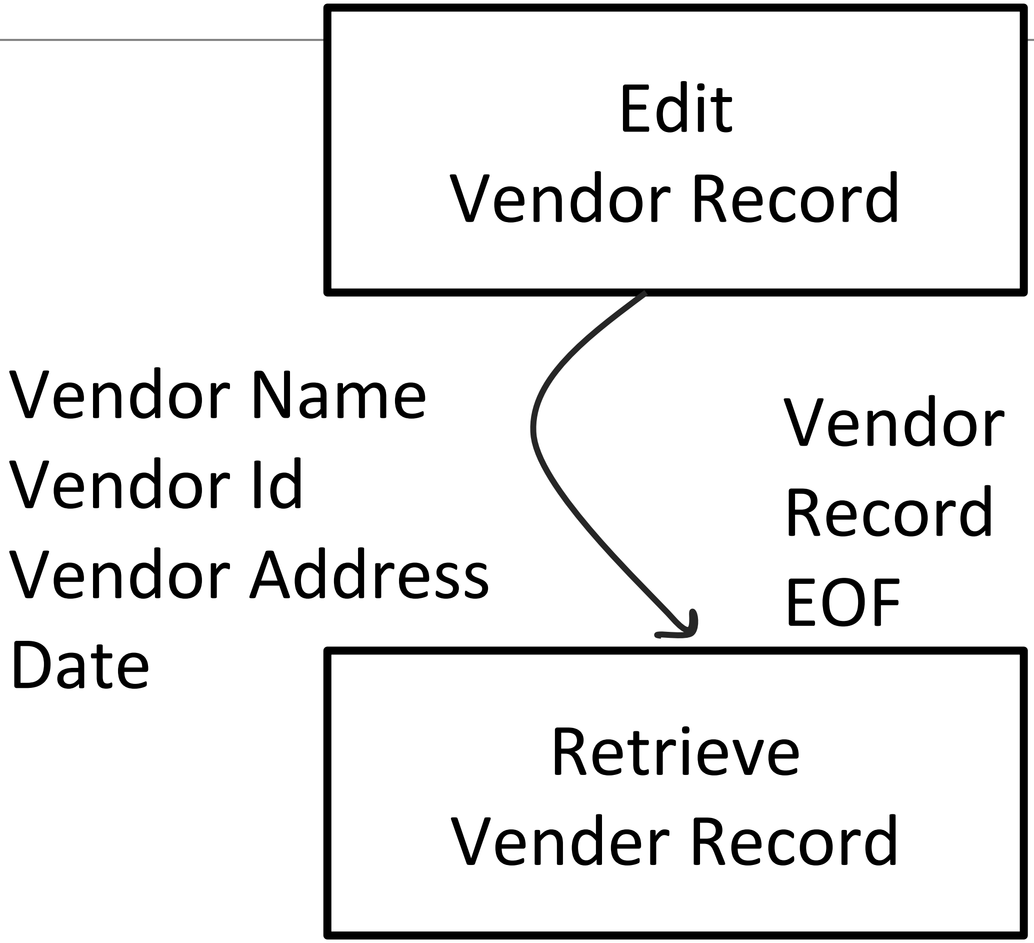
- High Coupling
- Low Coupling

High Coupling / Tight Coupling

These type of systems have interconnections with program units dependent on each other.

Changes to one subsystem leads to high impact on the other subsystem.

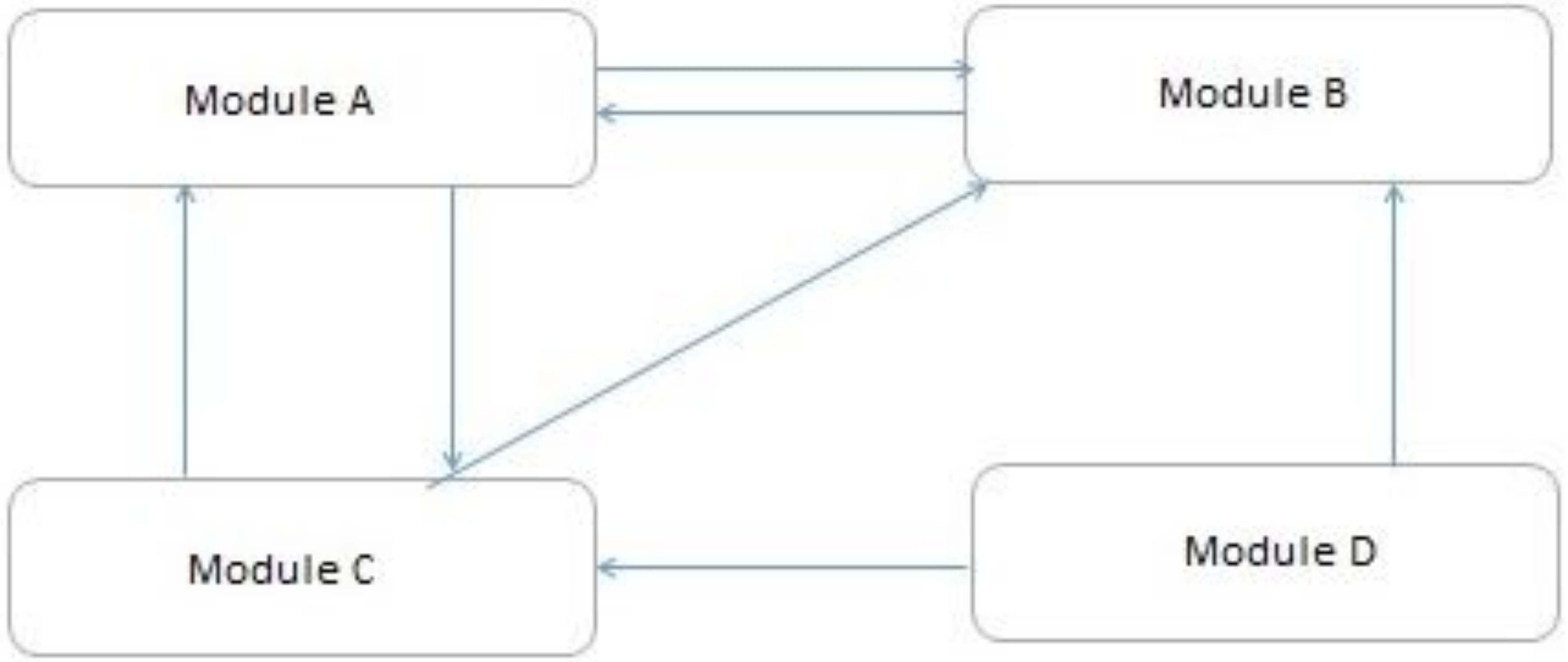

Module A
Module B
Module C
Module D

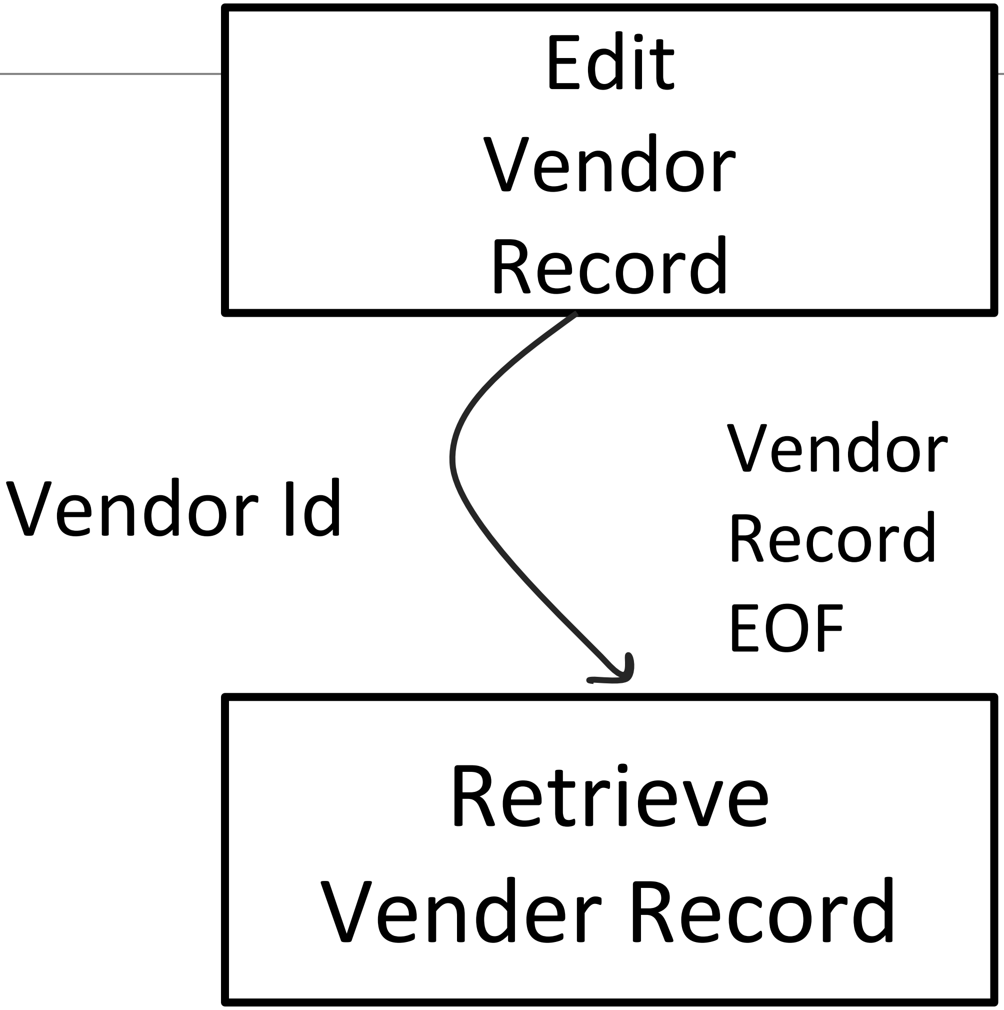

Edit
Vendor Record
Vendor Name
Vendor Id
Vendor Address
Date
Vendor
Record
EOF
Retrieve
Vender Record

Low Coupling / Loose Coupling

These type of systems are made up of components which are independent or almost independent.

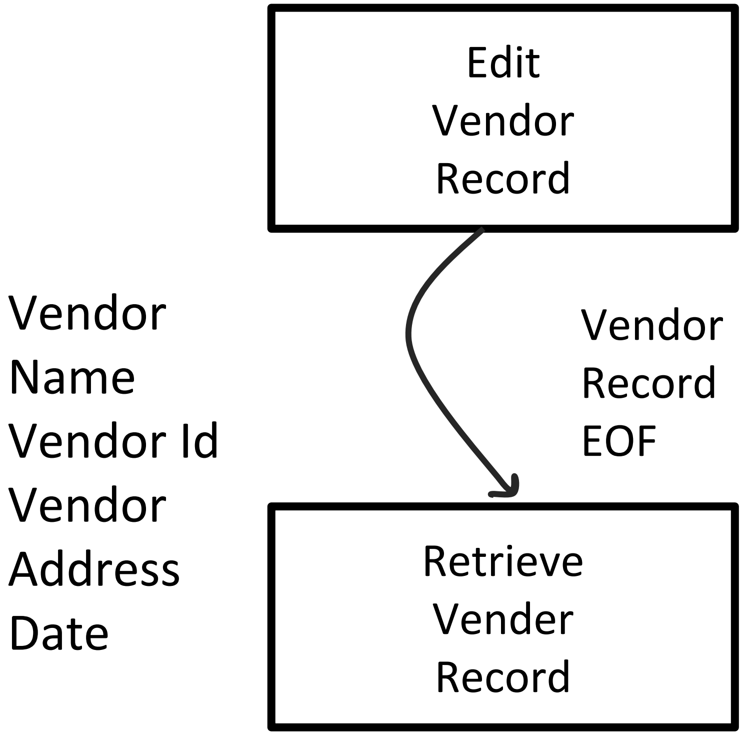
A change in one subsystem does not affect any other subsystem.


Module A
Module B
Module C
Module D


Edit
Vendor
Record
Vendor Id
Vendor
Record
EOF
Retrieve
Vender Record

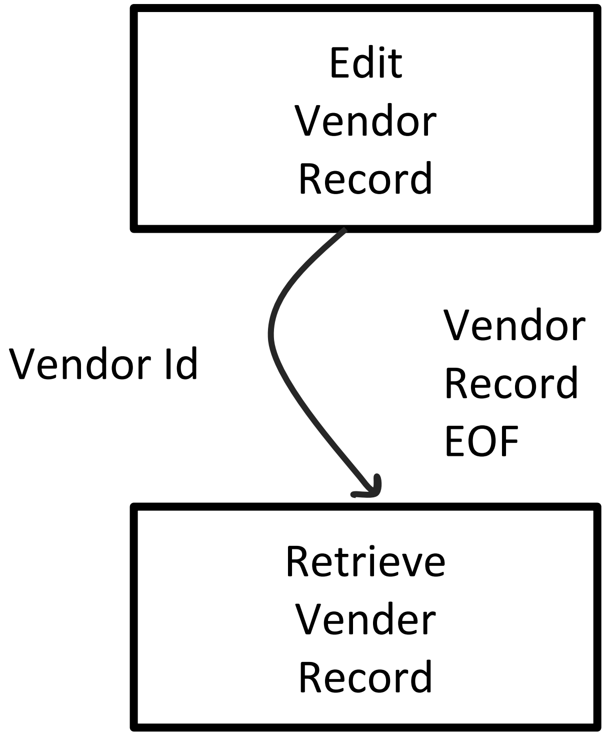
Advantages of Loose Coupling over High Coupling

- Loose coupling minimizes the interdependence between.
- Control the number of parameters passed between modules.
- Avoid passing unnecessary data to called modules.
- Pass data only when needed.
- Maintain superior / subordinate relationship between called and calling modules.
- Pass data , not control information.



Poor

Instead

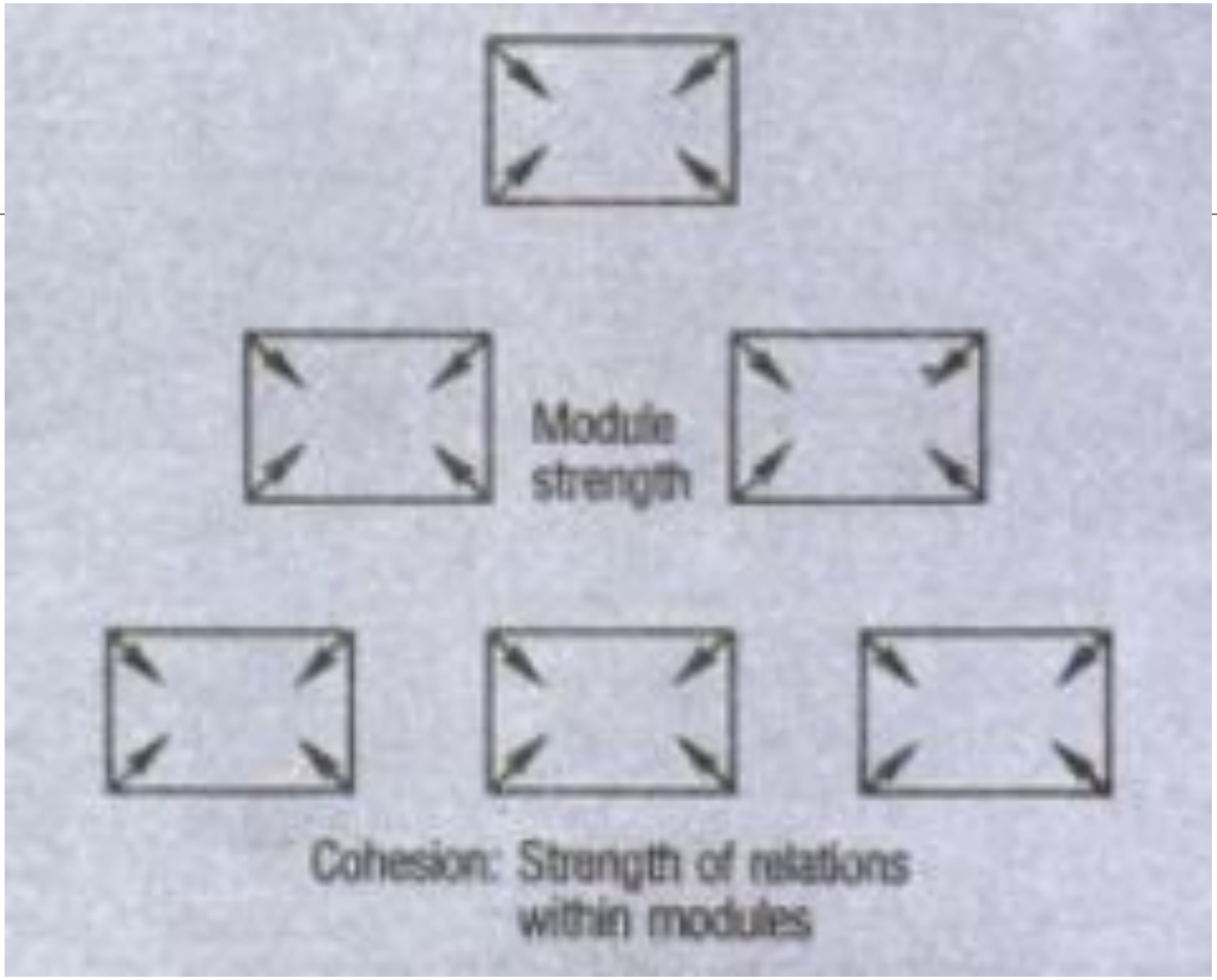


Good

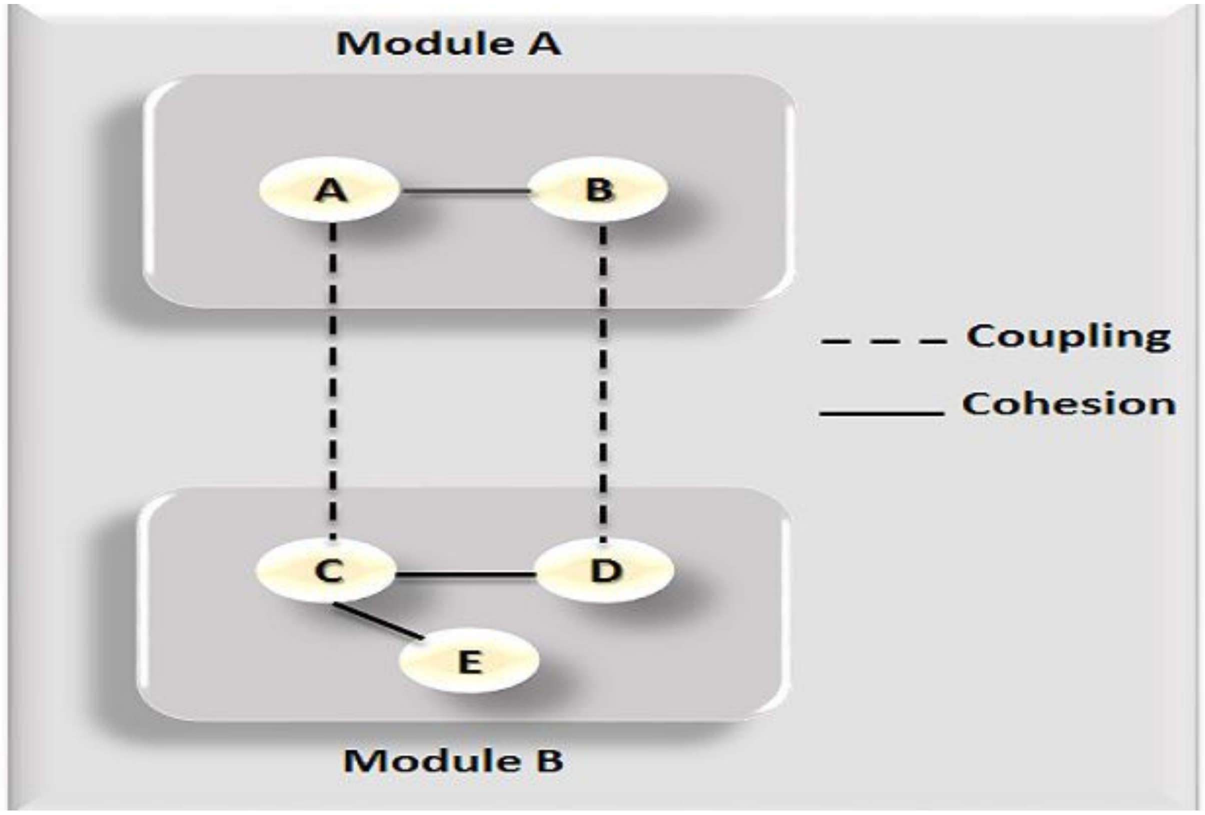
Cohesion

Cohesion is the indication of the **relationship within module.**

Cohesion shows the module’s relative functional strength. It defined as the degree to which all elements of a module, class, or component work together as a functional unit.

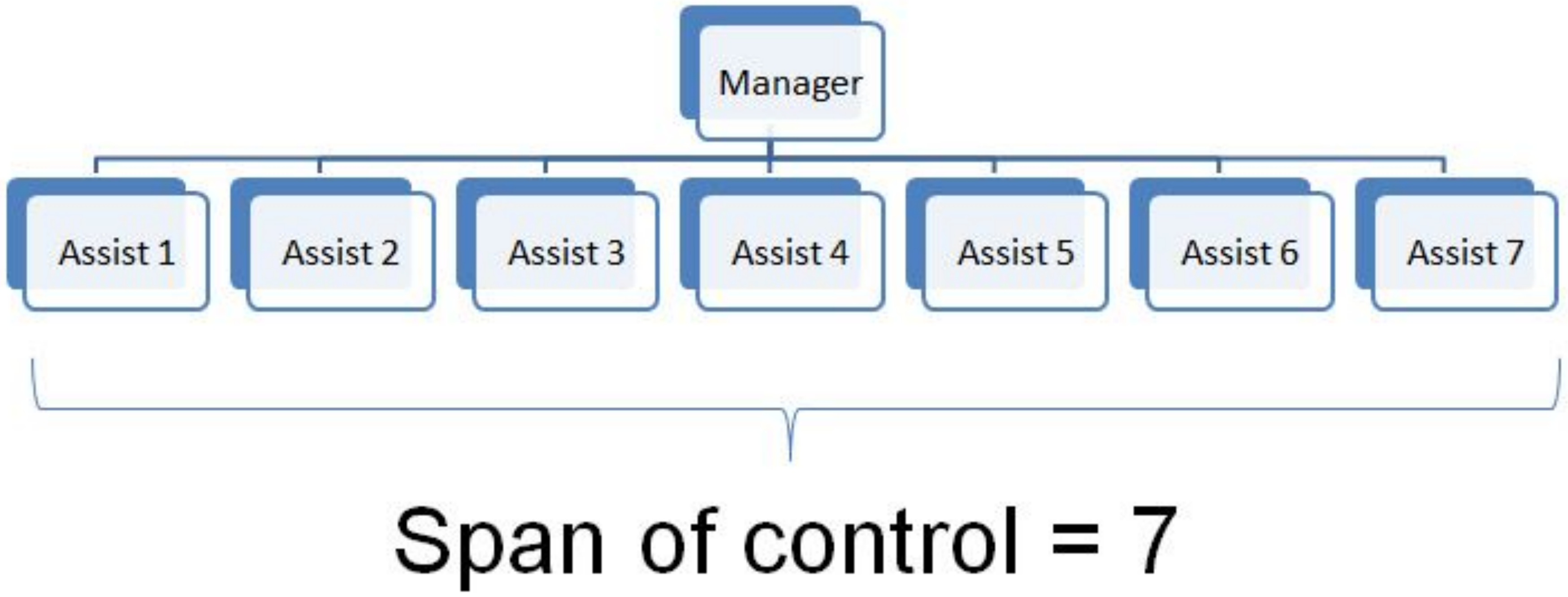

Module strength
Cohesion: Strength of relations within modules

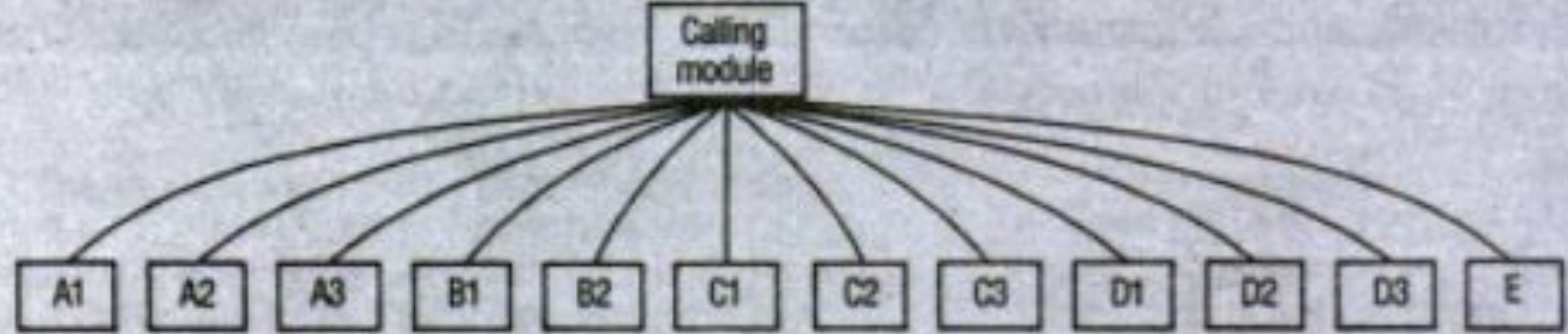
Coupling and Cohesion



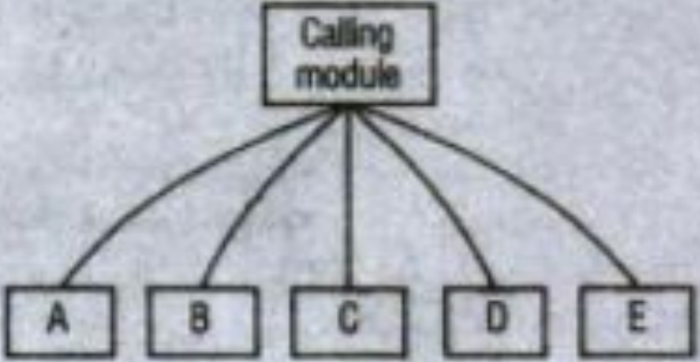
Span of control refers to the number of subordinate modules controlled by a calling module.

In general, we should seek to have no more than 5 – 7 subordinate modules.





Unacceptable span of control:
too many subordinate modules



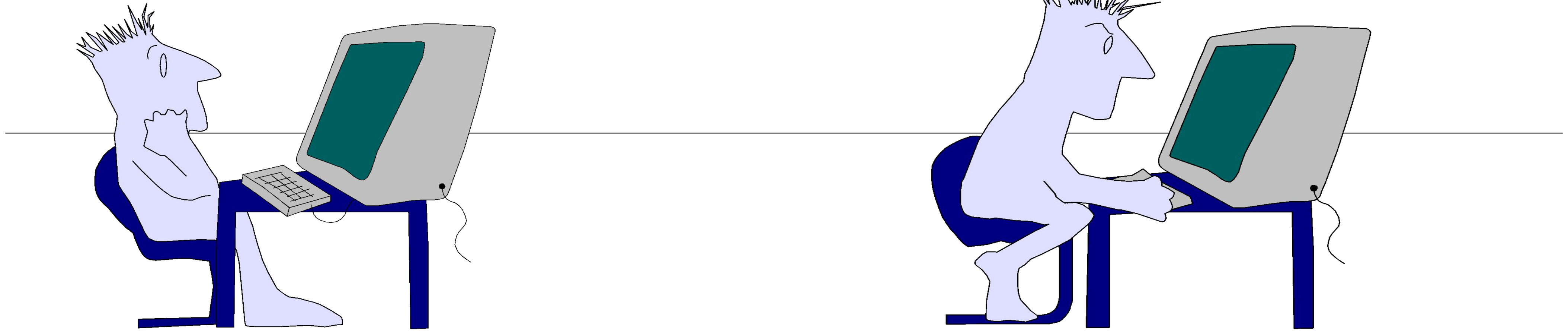
Acceptable span of control

Module size refers to the size of the module in terms of number of instructions. It will depend on the generation of the language (2nd, 3rd, 4th) and specific language within that generation (COBOL, C , Java).

Shared Modules refers to having one module do a job that is required frequently by number of modules and then sharing it (example data validation).

Software Testing

“Software testing is the process of finding defects in the software so that these can be debugged and the defect-free software can meet the customer needs and expectations.”



If developer tests the system

Understands the system
but, will test "gently"
and, is focused on "delivery"

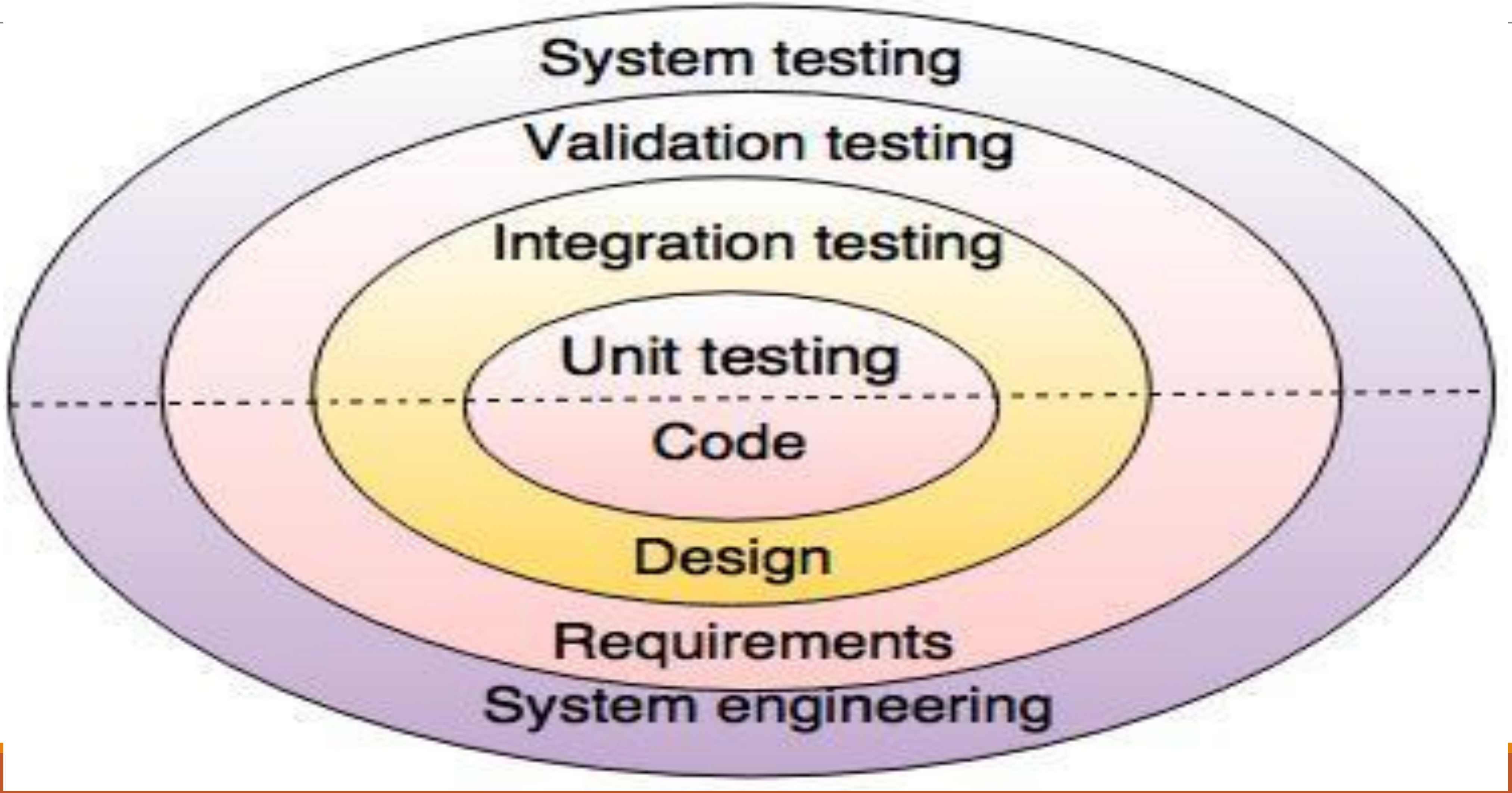
independent tester

Must learn about the system,

but, will attempt to break it

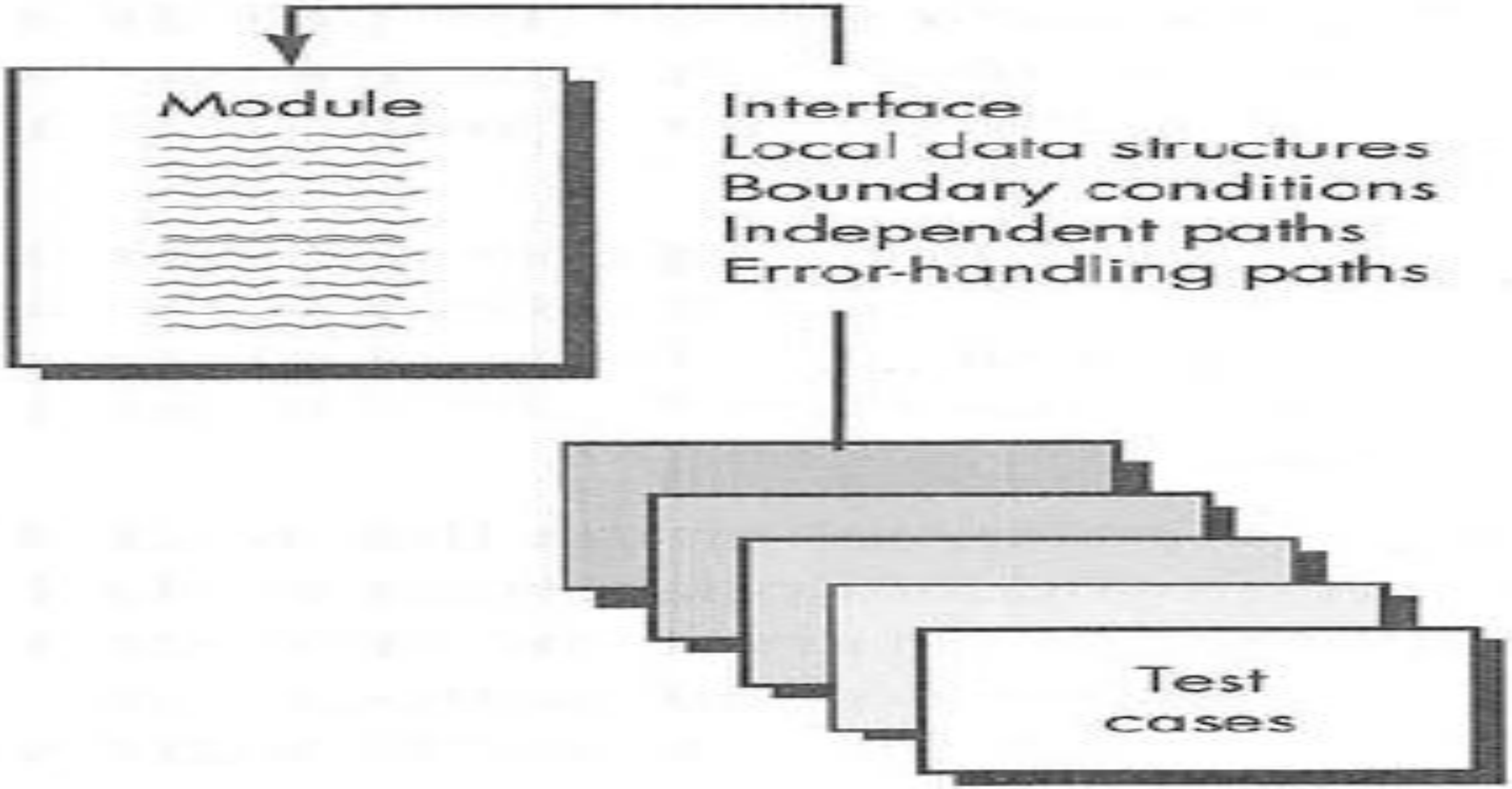
and, is focused on "quality"

Software Testing Strategies

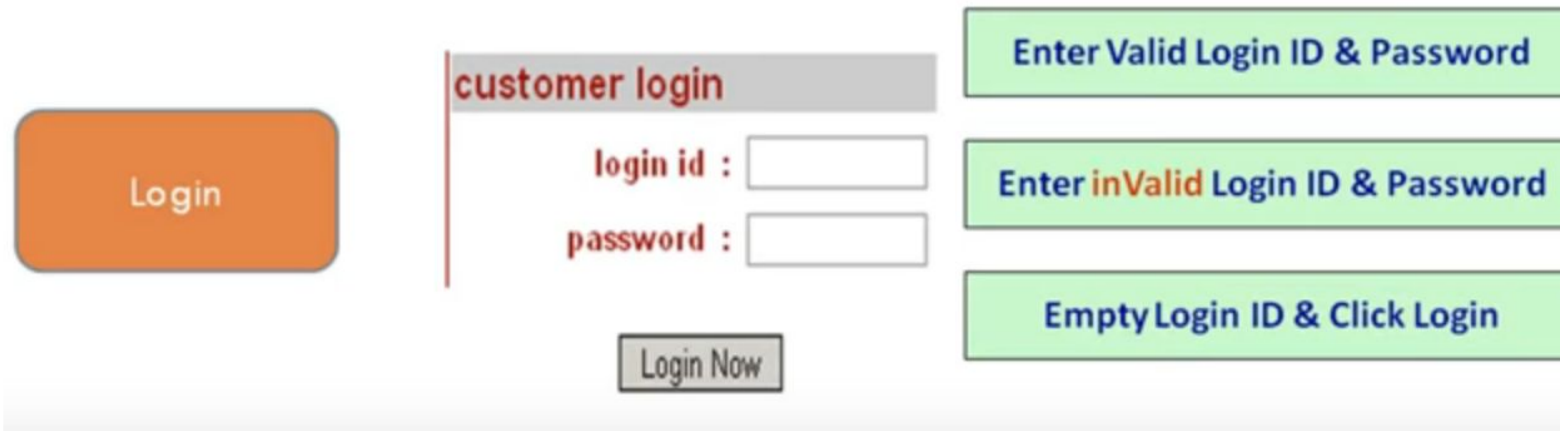


Unit Testing:

- **Unit Testing** is a level of the software testing process where individual units/components of a software/system are tested.
- The purpose is to validate that each unit of the software performs as designed.
- Unit testing focuses verification effort on the smallest unit of software design – the software component.
- **Who performs it?**
  - Unit Testing is normally performed by software developers. In rare cases it may also be performed by independent software testers.


Module
Interface
Local data structures
Boundary conditions
Independent paths
Error-handling paths
Test cases

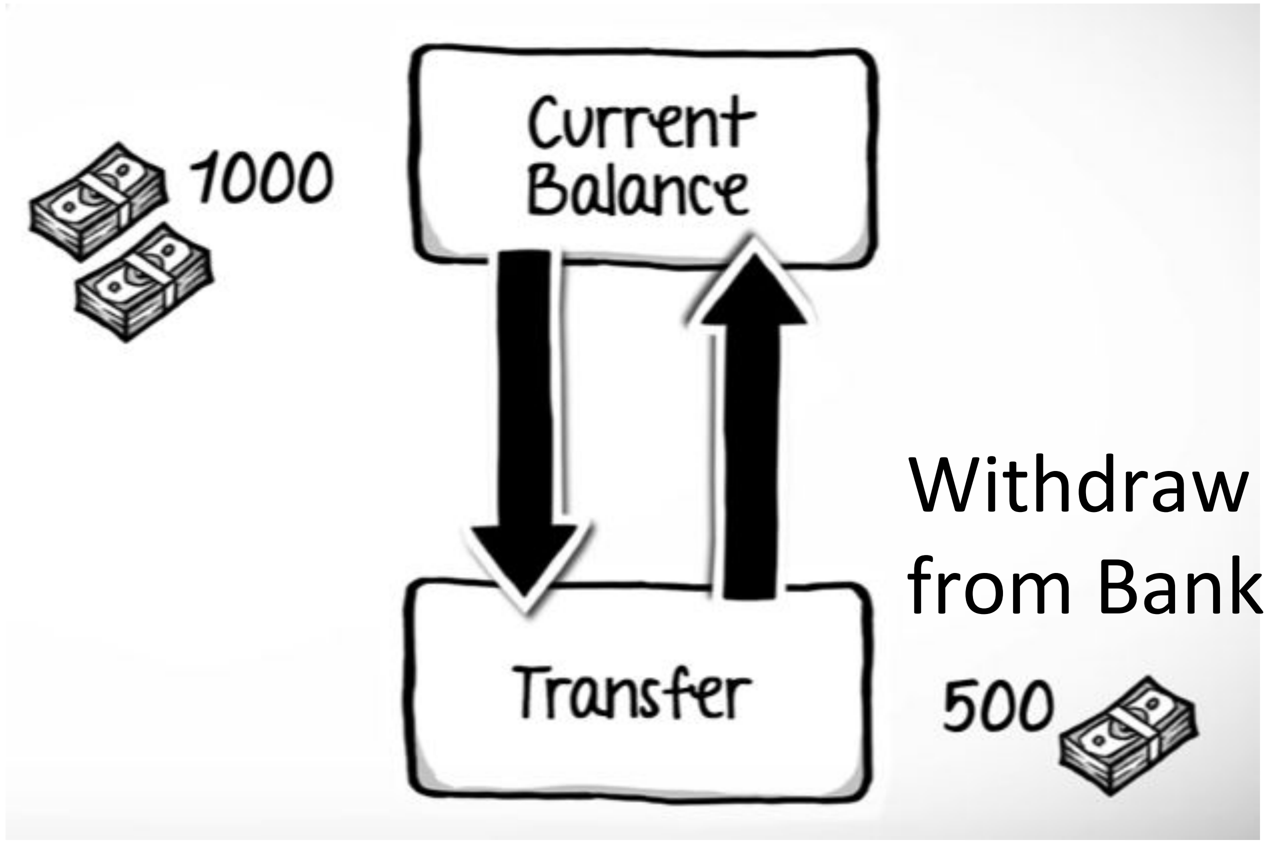
Example of Unit Testing with Login Page


Login
customer login
login id :
password :
Login Now
Enter Valid Login ID & Password
Enter inValid Login ID & Password
Empty Login ID & Click Login

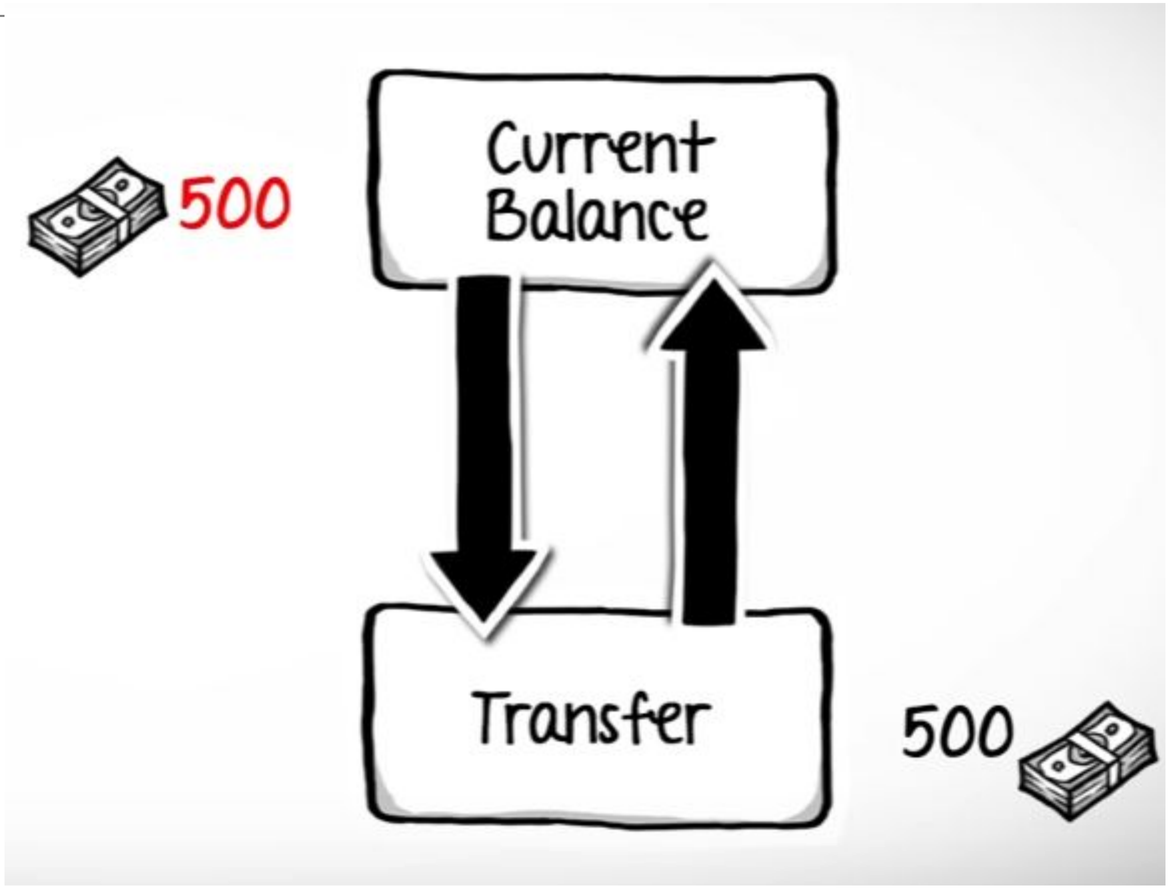
Integration Testing

- Once parts work properly we have to integrate them to get the whole system.
- But there is no guarantee that the integrated system will work properly since parts are working properly. That is why we need integration testing.

Example of Banking System



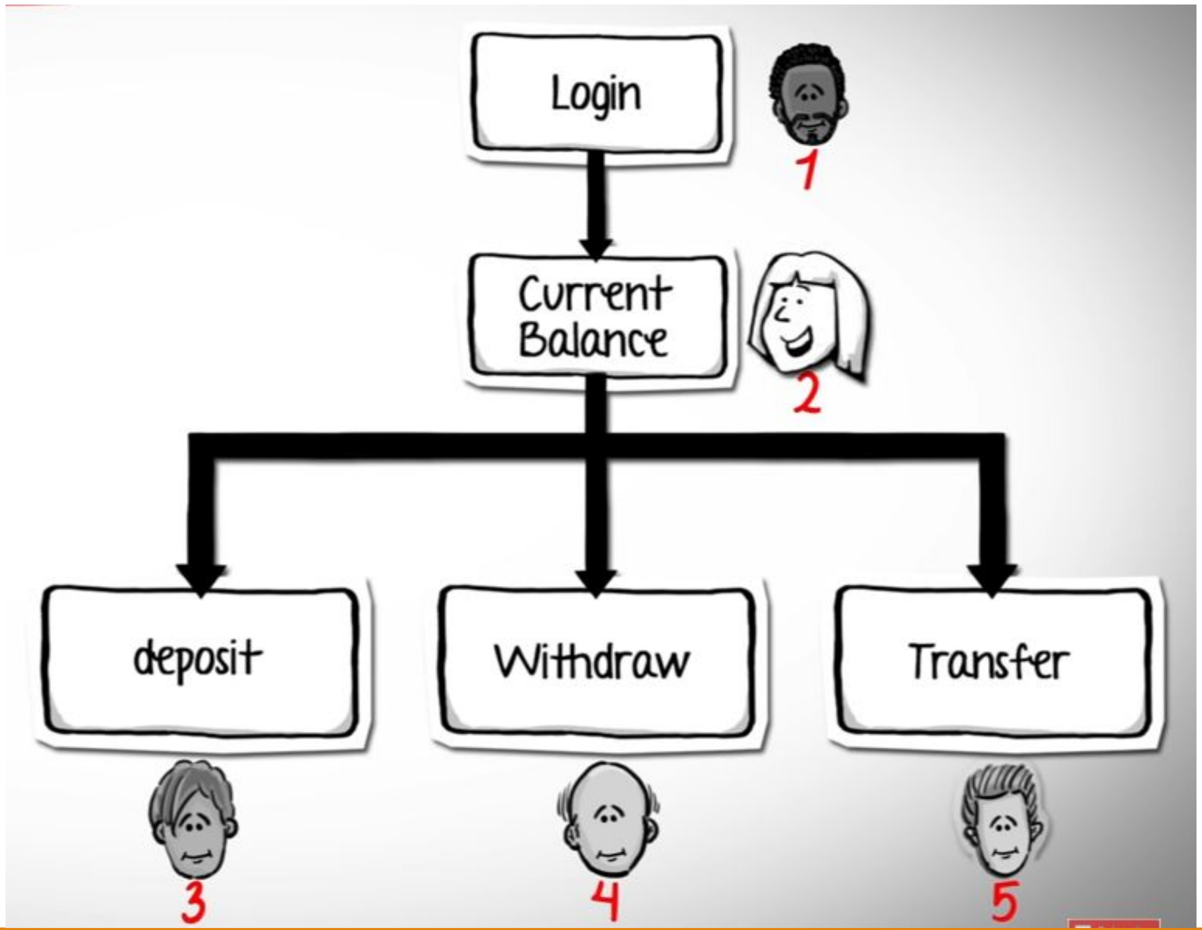
Withdraw from Bank



Types of integration testing:

- For integration there are two approaches:
  - “big bang” / non-incremental
  - Incremental

Big-Bang Approach



Tester have to Wait for all modules to developed.

It is time Consuming

If bugs are there it is difficult to trace root cause of bugs

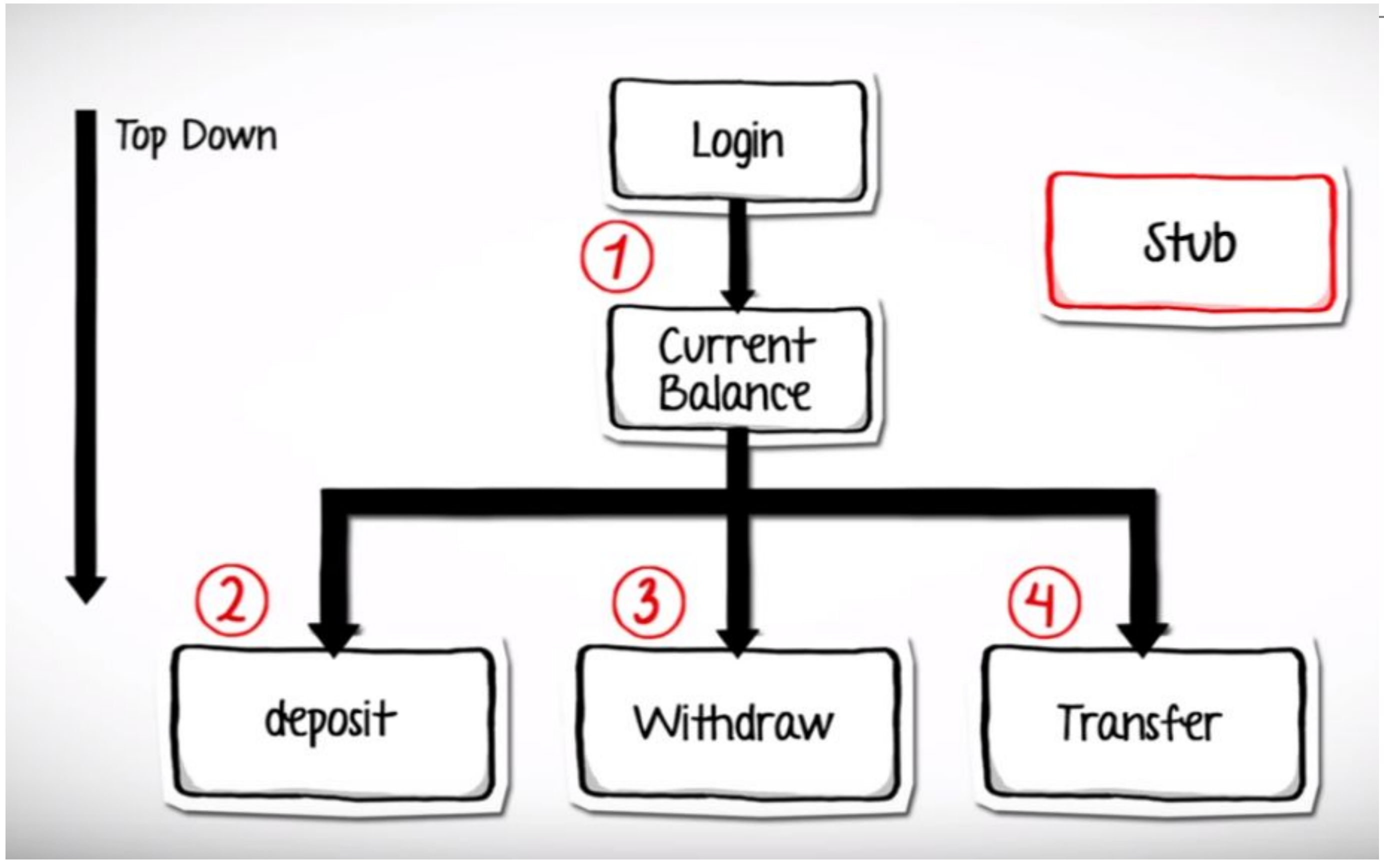
Incremental Approach

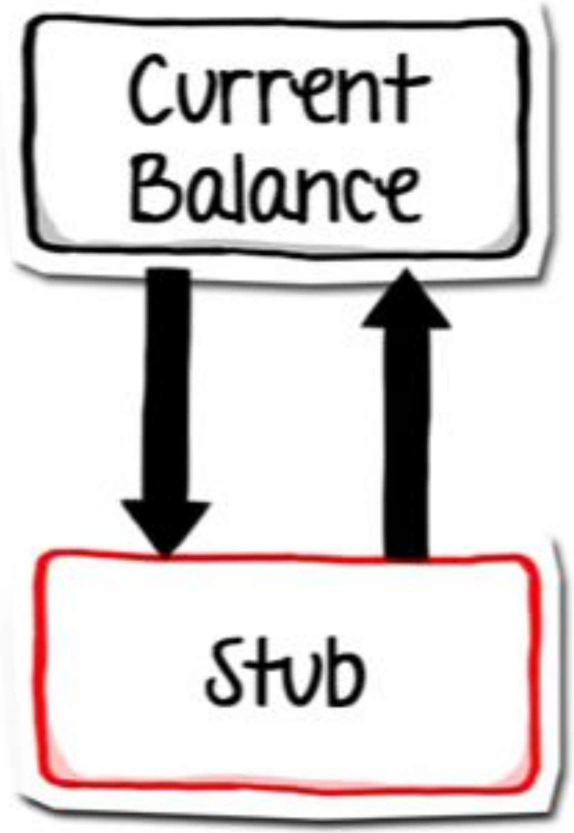
In incremental approach the program is constructed and tested in small increments, where errors are easier to isolate and correct and interfaces are tested more completely.

Current Balance

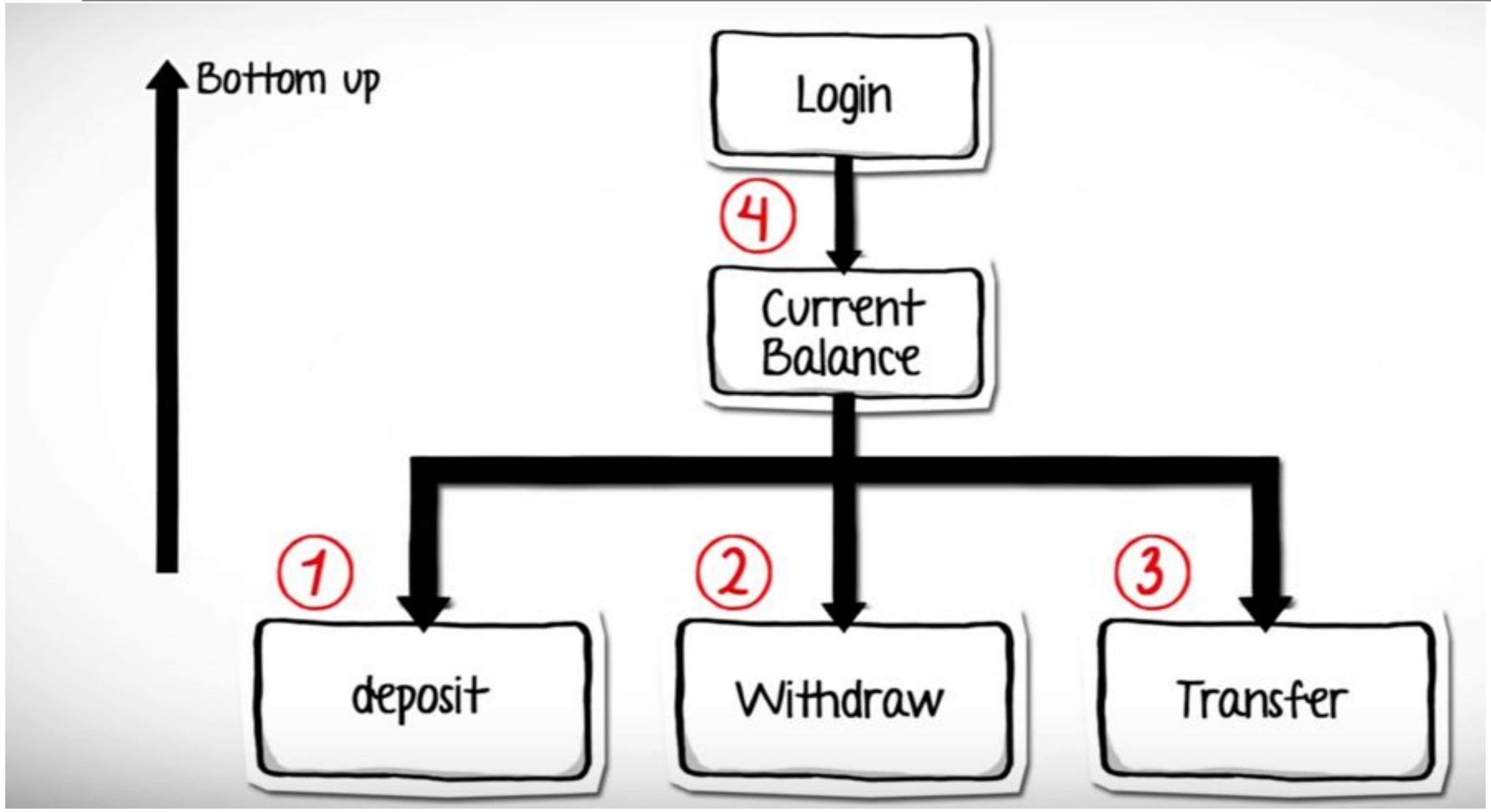
If current balance and transfer modules are ready then we can test those two modules.

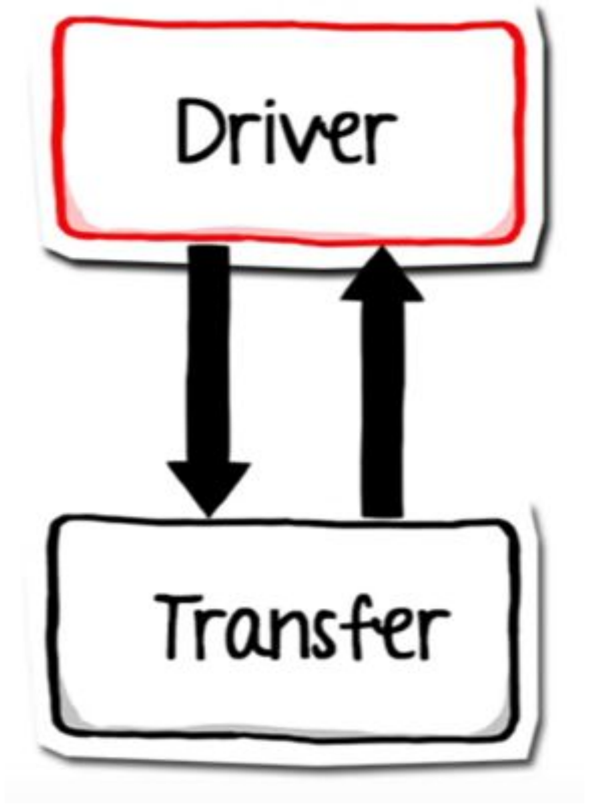
Top down approach: A module that can accept data or pass data that module is converted to Stub.





Bottom Up approach: A module that is calling modules that module is converted to Driver.





Validation testing

- This testing focuses on user visible action and user- recognizable output from the system
- Software Requirement Specification (SRS) describes all user-visible attributes of the software and contains a validation criteria section that forms the basis for a validation testing approach

System testing

- The software is tested with system elements as a whole. Types of system testing:

1) Recovery testing
2) Security testing
3) Stress testing
4) Performance testing
5) Deployment Testing

1) Recovery testing:

This testing refers to intentionally failing the system by creating a data loss event and then to see whether and how fast the recovery is possible.

This will test the recovery plans and procedures and may lead to corrections in the same.

2) Security testing:

It attempts to verify that protection mechanisms built into a system, protect it from improper access or entry in the system.

The tester plays a role of the individual who desires to access the system.

3) Stress testing:

Stress tests are designed to confront programs with abnormal situations

Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency and volume

4) <u>Performance testing</u>:

- Performance testing is designed to test the run time performance of s/w within the context of an integrated system

<u>5) Deployment Testing:</u>

- In many cases, software must execute on a variety of platforms and under more than one operating system environment.

Designing test data

~~There are two very different sources of test data, live and artificial. Both have distinct advantages~~ and disadvantages for the user.

Live test data are those that are actually extracted from organization files. This will test the system for its normal operation. Since this is typical data it does not test all the combinations and the bias towards the typical data does not provide a true system test. It ignores the cases most likely to cause system failures.

Artificial data are created solely (only, exclusively) for test purposes. They can be quickly prepared by using a data generating utility program and make possible the testing of all logic and control paths through the program.

Unit 4
Managing System Implementatio

Introduction

- Analysts responsible for implementation must pay attention to every minute (small, tiny) detail.
- There are three main aspects of implementation.
  - Training personnel
  - Conversion procedures
  - Post-implementation review

Training methods

- The training of operators and users can be achieved in several different ways.
- The training activities may take place at vendor locations; at rented facilities (hotels/training institutes/universities); or in-house at employees' organizations.

Vendor and In-service training

- As the name suggests this training takes place at vendor location.
- The equipment may be specially kept for training and hence there may not be rush to finish off training fast so that it can be utilized for production purpose.
- Sometimes this training is provided by the vendor without any extra charge.
- In case of special software like RDBMS or ERP package sending personnel to off-site short term courses providing in-depth training is preferable to in-service training.

- There is an additional advantage of interaction with users from other organizations and sharing of questions, problems and experiences with them.
- The only disadvantage is that it involves additional time and costs for travel to other cities if they are not offered in your own city.

In-house training

- As the name suggests this is the training given to the users in their on organizations (also called on-site training).
- The advantages are that the training can be personalized to the organization’s needs and setting, fees can be negotiated and made more economical and the organization can involve more people since traveling may not be required.
- The biggest disadvantage is that the employees are in their own organization and hence there may be distractions like telephone calls and emergencies.

Conversion

- Conversion is the process of changing from the old system to the new one. The analyst needs to know the methods of performing system conversion and the procedures used to ensure that it is performed properly.
- There are four methods of handling a system conversion.
- Parallel systems
- Direct cutover
- Pilot approach
- Phase-in method

- **Parallel Systems**
- The most secure method of converting from an old system to a new system is to run both systems in parallel.
- Users continue to operate the old system in the usual manner but they also start using the new system.
- This method is the safest approach, since it guarantees that in case of any problem in the new system the organization can still fall back to the old system without loss of time, revenue or service.
- The system costs double, since there are two sets of systems costs.

- **Direct cutover**
- The direct cutover method converts from the old to the new system abruptly (quickly, immediately), sometimes over a weekend or overnight. The old system is used until a planned conversion day, when it is replaced by the new system.
- There are no parallel activities.
- If it is absolutely necessary to go for the new system then this approach should be used. Psychologically it forces all users to make the new system work since they do not have the old system to fall back on.

- **Pilot approach**
- In this method a working version of the system is implemented in one part of the organization such as a single work area or module.
- The advantage of this method is that it provides a sound proof before full implementation.

- **<u>Phase-in method</u>**
- The phase-in method is used when it is not possible to install a new system throughout the organization all at once.
- The conversion of files, training of personnel, or arrival of equipment may force the staging of implementation over a period of time ranging from weeks to months.
- Some users will start taking advantage of the new system before others. Example : Bank computerization.

Conversion Plan

- The conversion plan includes a description of all the activities that must occur to implement the new system and put it into operation.
- List of files for conversion
  - Identify all data required to build new files
  - List all new documents and procedures that go into use during conversion.
  - Identify all controls to be used during conversion.
  - Assign responsibility for each activity.
  - Verify conversion schedules.

Site Preparation

- Very little site preparation is needed for installation of one or few PCs.
- But, if the information system is to be implemented with a powerful server and number of PCs connected in LAN, MAN or WAN then extensive site preparation will be required.

- Post-implementation review
- Post-implementation is to determine how well the system is working, how it has been accepted and whether adjustments are needed. The post-implementation review is also important to gather information for the maintenance phase of the system which is the first source of information.
- The analyst has to see the quality of system's output, the ease of use and tendency toward errors in input. User confidence is also an indicator of system quality. If it is low, the analyst must determine why it is low.