CS504 Software Engineering – I Lecture Wise Questions and Answers For Final Term Exam Preparation by Virtualians Social Network

Lec : 22

What is Software architecture?

It provides an over all view of a high level system.

In it, we define how a system is decomposed into subsystems,components modules and relationship among those components and subsystems etc.

What is the importance of software architecture?

It serves as a mean of communication between the stack holders.

Helps in making early design decisions.

Reusability is achieved.

Attributes of software architecture address both functional and non functional requirements of a software system. These attributes are as under:

Performance → by localizing the operations to minimize the communication between subsystems.

Security→ by adopting a layered architecture.

Safety→ by isolating a specific/safety critical components.

Availability→ by building redundancy i.e. by providing suitable alternatives in case of an expected or unexpected system failure.

Maintainability→ by using self contained components i.e. change in a component should not bring a change in another component.

Lec 23:

A software Architecture usually consists of the following elements which are presented in a model below :

Logical view/Functionl view→ comprises of various different functions provided by the system abstraction and the domain elements.

Development View→ Comprises of the different files and directories in the system.

Code view→ Comprises of classes, objects, procedures function, subsystems, layers and modules.

Concurrency View→ Documents different parallel processes and threads in the system having its main focus on even synchronization and parallel data flow.

Physical view→ Depicts how a system would physically be deployed.

Architectural styles may describe → i) A set of components that perform a function required by a system. ii)A set of connectors that enable communication among components.iii) Integrating components from the system and semantic models that enable a designer to understand the over properties of a system.

what is control modelling?
Architectural design process consists of three activities; System structure, modular decomposition and control modeling. These three activities are performed to design architecture of a system.
In system structuring, we divide system into major blocks (sub systems). In modular decomposition, these sub-systems or blocks are further decomposed into modules. These modules should be connected in term of control flow to make system executable. This is done through control modeling. In control modeling, we model flow of control among different modules of the system to enable different sub-systems to communicate and for working as a single system.

In client server model why server does not need to know about clients? please explain

Normally the total no. of servers in a organization remain fix whereas the clients go on increasing and decreasing. If we implement the system in a way, where server doesn't know about the clients, then this gives us the flexibility to add/delete clients easily. In this case adding a client mean just connect the client with the server and start using the services.
Also in most of the cases server don't need to know about the clients, so keeping record of clients may be burden especially in case when the clients increase significantly.

Lec 24:

Architectural Styles:

Data centered→ shared data is held in a centeral data base or each subsystem maintains its own database.

Client server model→ a distributed system model which describes how data and processing is distributed across a range of components. Application is modeled as set of services provided by the servers and a set of clients that use these services. Clients know the servers but the servers do not need to know all the clients. Mapping between processors is also not always 1:1.

Server types→ File servers, Data base servers, Transaction servers, Groupware servers.

Server Configuration:

Thing client Model→ Client machine has no functionality on it. It just provides the user interface.

Fat client Model→ more processing is delegated to the client as the application processing is locally extended.

concept of architectural views, specialy logical view please sir elaborate it in simple words and it is also request that define it with the help of any real life example.???

Please note that an architectural views represent the viewpoints of different stakeholders. That is a view presents system from one angel, other view presents system from another angel and so on. eg, just as a building architect might create wiring diagrams, floor plans, and elevations to describe different facets of a building to its different stakeholders (electricians, owners, planning officials), so an IT architect might create logical view, implementation view and so on.

In logical view, architect presents the system from end user point of view that what functionality and services the system provides. It describes how the system is structured in terms of units of implementation. Normally this view includes class diagram, communication diagram, sequence diagrams etc.

An Architecture style is a set of principles which you can use to build a system, normally these principals are re-used in applications, e.g. client/server architectural style, component based,

layered etc etc. each of which provides a way to implement software structure.

Non functional requirements are qualities and characteristics of a system. They define the criteria to judge the operation of the system. e.g. system must be maintainable, system must be secure, system must be user friendly etc etc.

Lec 25:

Zero Install→ Where there is no or little processing done at the client side. It differes from Thin client model in a way that in case of thin client model, the entire processing is done by a single server, whearas in case of zero install, the network envoirnment is used to distribute server side processing by adding a number of servers which share processing load.

N-Tier→ Here, the idea is to enhance the scalability and performance by distributing both the data and the application using multiple server machines.

3-Tier→ each application architencture layers may run on separate processors. These layers include presentation, application and data base layers. In case of N-tire architecture, the system may have more than 3 layers.

Data Flow/Pipe architecture→ Used when the input data is processed through aseries of transformations to yield the desired output. Each filter works independently others.

Layerd Architecture→ Each layer isolates the outer layer from the inner complexities. The outer layer only needs to know the interface provided by the inner layer.

Reference Architecture→ It is only a reference for defining protocols and designing and implementing systems developed by different parties.

Partitioning the Architencture→ To distribute the responsibilities to different subsystems so that we get a software system which is easy to maintain. Partitioning may be horizontal or vertical.

Horizontal→ separate branches of the module hierarchy for each major function and control modules are used to co-ordinate communication between functions.

Vertical→ The architecture is partitioned in horizontal layers having decision making modules residing at the top and worker at the bottom.

The standard and mostly recognized pattern to be followed for SRS, is that specified by IEEE. So why should anyone follow some other pattern dictated by some other GANG...GOF?
Yes. Standard is defined by IEEE. where GOF is collection of design pattrens which can be followed. But these too, does not restrict the design pattren choice. Choice of design pattren is dependent upon requirements of a system. Even if IEEE standard is not followed, it does not make harm. Design pattren can be chosen from GOF and even design pattren can be hybrid or customized depending upon need.

What is meant by Frame Work? and what is an integrated Frame Work? is it same as famous Microsoft .Net Frame Work? how do design patterns apply to frame work?
Framework is a a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful. For example, .NET framework is a framework used to develop, deploy and to run applications by using provided services, libraries and modules. Design pattrens represent solutions to problems that arise when developing software within a particular context. Patterns can be characterized as more abstract descriptions of frameworks, which are implemented in a particular language.

importance of documatation?? also good documentations techniques.?????

Documentation is very important. Work done at each phase is required in the next phase. For example, requirements are mapped in design so work done at requirement phase is required in design phase. Design is required to convert into implementation. Best way to convey previous work done is through documentation. So at one hand, documentation is required for well managed development. On the other, it is required for later maintainenace. When system is in use, some problem occurs, maintenance team requires documentation to identify exact place of problem as well as for measuring the impact of change on other parts.
This is the reason that why documentation is needed to be focused. There is no specific technique for documentation. Documentation is done in parallel with development. At every stage, everything is documented. For example, in requirement phase, SRS and FRS are developed. In design phase, design document is developed containing every model. In testing phase, test-cases are written. Document of a project is collection of all these documents. They are merged into a single document to provide complete information about system.

Lec 26:

Design Patterns ◊ are the devices that allow programs to share knowledge about their design, such knowledge/ solutions can be used repeatedly over time.
GOF (gang of four) Design pattern format includes :

Name, problem, context, forces, solution , result, rationality , related patterns and known uses of the format being applied .

Patterns can be classified as under :

According to their creation ◊ here, we look into how the patterns are created.

According to structure ◊ here we deal with the object's nature.

According to behavior ◊ here, we use inheritance to distribute behavior between classes.

Framework :

A framework simply dictates the architecture of the software. In other words, framework consists of re-useable patterns.

Best Regards from pp !

Can we say that the OSI reference model represents horizontal partitioning of a system?

OSI reference model is a standard reference model for communication between two end users in a network. Layers of OSI model works on different layers of network and enables user to communicate. Where horizontal partitioning is a way to organize modules based on functionality. Such partitioning eases in later extension with least effect on other modules. So we can not say that OSI model represents horizontal partitioning because horizontal partitioning is work being done at application layer where OSI model is meant for working on all layers of a network.

Let's suppose there is a system which comprises of following servers connected via network:-

1. Web Server
2. Email Server
3. Security Server
4. Database Server

A client which wants to access the web-server, can directly access the web-server by http protocol....

However, Will the client also access email server directly or will it be indirectly through web-sever? Similarly what will be the case for database server?

Database server, web server, email sever etc are always remotely located. When we talk about network that consists of a number of clients. There is always a main server (gate-way server). Every request of client is directed towards that server. Server will direct each client accordingly whether clients wants to connect with email server, web server or database server. Gate-way server is a sort of traffic controller in network. An other reason of directing data through server is locational transperacy and security servers as well as of network.

Lec 27:

Observer pattern / Creational :

It basically relates to a many to many relationship between objects so that when on object changes state,all of its dependants are notified and updated accordingly.

It is used when multiple displays of state are needed or in other words, when data is to be presented in more than one way.

Singleton pattern / Behavioural :

It basically makes sure that a class only has one/few instance/s of a class and it must be accessible to clients from a well know access point.

Façade pattern / Structural :

It simply provides a unified interface/door-way to a set of interfaces in a subsystem. It defines a higher level interface that makes a subsystem easier to use.

Regards from Persian !

Software architecture just represent non-functional and functional requirements? Is it represent Business requirement and user requirements ? Kindly explain that which thing comes under software architecture...???

Software architecture defines the high level structure of the software by putting together a number of architectural elements in an organized fashion. These are: data elements, processing elements, and connecting elements. These elements are chosen to satisfy the functional as well as non-functional requirements of the system.
Software Architecture does not represent non-functional and functional requirements but a way to achieve/satisfy these requirements. A software system is built by focusing on all requirements. When we say non-functional and functional; we are covering every possible requirement of a system. If a system is performing all required functionalities, that means we have achieved business requirement and we have also achieved user requirements as functionalities are for users.
Software architecture covers all design and architectural decisions which are related to data, processes and connecting elements.

How can we differentiate main frame based application, thin client model and zero install? As all three have many similarities.??

As you have studied in your handouts that in any system, sub-systems may need to exchange information and data.

There are two ways in which it can be done:

1. Keeping the data into a central database or repository and may be accessed by all subsystems
2. Each sub-system maintains its own database and passes data explicitly to other subsystems

For large main frame based applications, we use the first approach for sharing the data among different subsystems. The Data-Centered architecture have the goal of achieving the quality of integrability of data. The term refers to systems in which the access and update of a widely accessed data store is their primary goal. Basically, it is nothing more than a centralized data store that communicates with a number of clients. There is a software installed on each of the clients connected in the system and using this installed software, the client may send, share or retrieve the data with any other client in the system.

In case of Zero-Install, nothing needs to be installed or put into client. This makes system maintenance an easy task. Whole processing will be carried out by the server which may affect the performance of the system in case of a single server. To overcome this problem, different processing responsibilities are distributed among number of servers. The major difference between Data Repository Model and Zero-Install is the availability of number of servers required for the processing. Data Repository is comprised of such a single data server where as in case of Zero-Install; different processing responsibilities of the system are distributed among number of different servers such as database server, application server, web server etc.

What is difference between terminals and computers? Kindly explain the differences.??
Terminal is client computer. Consider a network in which there is a sever and 100 client computers. These client computers are terminals. When you are working on your computer at home, that is not terminal because it is not connected to any network. But when you are working in VU lab, your computer is terminal of network. So in client-server model based network, every client is a terminal node of network.

is uml include out documentations??

Unified Modelling Language (UML) is a design methodology that aids us in designing the software. It is part of design documentation which is ultimtely added in final documentation. UML is a part of documentation.

what is main work of facade ?

Answer :
The facade pattern is a software design pattern commonly used with object-oriented programming. A facade is used when one wants an easier or simpler interface to work with.
A facade can:
1) Make a software library easier to use, understand and test, since the facade has convenient methods for common tasks;
2) Make the library more readable, for the same reason;
3) Reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system;
4) wrap a poorly-designed collection of APIs with a single well-designed API.

Lec 28:

Maintainability of a code means that the code should be clear, simple and flexible.

Self documenting the code means that the code should explain itself without the need of comments and extraneous documentation.

Naming conventions are 12 whereas specific naming conventions are about13 which you can read and get idea about in the handouts.

Summing it up, the code should be user friendly. And this can be achieved only through adopting simplicity, clearness, flexibility and conventions.

Regards from Persian !

explain the N-Tier architecture ?

Answer : In n-tier architecture, in order to increase performance, we may distribute the application over different servers by putting different subsystems on different servers. In n-tier architecture, the presentation, the application processing, and the data management are logically separate processes. It is also possible that different components of presentation or database may be put on separate servers. For example, in case of three-tier architecture, there are normally two servers that is, the database server and application server but in case of N-tier architecture, each responsibility of the system such as database or logic may be put on more than one servers.

These kinds of architectures are used in the situation in large system where the main concern is to maintain the system efficiency.

how we can differentiate with thin, flat, zero model ?

Answer : In case of Thin Client Model, GUI are implemented on client side, whereas all the system processing is done by Server. It means that all the powers are assigned to server and in this case, the client only sends a request and server acknowledges it after doing some processing. Thin-client computing is also a way of easily maintaining computational services at a reduced total cost of ownership. Thin-client computing is also a way of easily maintaining computational services at a reduced total cost of ownership. Virtual Network Computing (VNC) is a graphical desktop sharing system which uses thin-client model.

In Fat Client Model, some of the processing responsibilities are also assignment to Client which reduces the burden over the server. For example, in this model, the application is installed on client side, making a distributed processing model where responsibilities are shared among client and server components of the system. But there is some disadvantage associated with this model. That is, if there is any need for application modification or updating, it is needed to be done on all the clients in the system which increases the maintenance effort. This client models are suitable for multiplayer video gaming and many other things. Most PCs (personal computers), for example, are also fat clients because they have their own hard drive DVD drives, software applications and so on.

Due to the maintenance overhead associated with Fat Client Model, it is difficult to implement such model and hence the solution it to adopt Zero Install Model. As oppose to Fat Client, nothing is installed on client side. Web-based application where web-pages are put on a web-server is an example of this type of architecture.

Lec :29

File handling tips for Java and C++

1. C++ header files should have the extension *.h*. Source files can have the extension *.c++* (recommended), *.C*, *.cc* or *.cpp*.
MyClass.c++, MyClass.h

2. Classes should be declared in individual header files with the file name matching the class name. Secondary private classes can be declared as inner classes and reside in the file of the class they belong to. All definitions should reside in source files.
3. Special characters like TAB and page break must be avoided.

4. The incompleteness of split lines must be made obvious.

Include Files and Include Statements for Java and C++

1. Header files must include a construction that prevents multiple inclusion.

Statements in Java and C++

1. Type conversions must always be done explicitly.

   floatValue = (float) intValue; // NOT: floatValue = intValue;

2. Types that are local to one file only can be declared inside that file.

3. The parts of a class must be sorted *public*, *protected* and *private*. All sections must be identified explicitly.

There are many more tips like that in the handouts so I suggest you should have a look at the handouts as well.

Regards from Persian !

what do you ment by application server.

An application server is a server that provides software applications with services such as security, data services, transaction support, load balancing, and management of large distributed systems.

what do you ment by web server

Ans

Web server can refer to either the hardware (the computer) or the software (the computer application) that helps to deliver Web content that can be accessed through the Internet.

Lec 30 :

Comments : In general, the use of comments should be minimized by making the code self-documenting by appropriate name choices and an explicit logical structure.
Expressions and Statements

Basic indentation should be 2.
for (i = 0; i < nElements; i++)
a[i] = 0;

Expression should be written as if they are written as comments or spoken out aloud.

Parentheses should always be used as they reduce complexity and clarify things by specifying grouping.

Complex expressions should be broken down into multiple statements. An expression is considered to be complex if it uses many operators in a single statement.
Sometimes the programmers, in their creative excitement, try to write very concise code by using shortcuts and playing certain kinds of tricks. This results in a code which is cryptic in nature and hence is difficult to follow.

In Lec-19 we studied about how to make structure of a system. My question is what is the purpose of defining inheritance relationship? I mean in example given Cash Payment, Cheque and Charge are child classess of Payment. Why I wrote Payment class in the first place, in system I will be using objects of either Charge, Cash payment or Cheque; then why we exclusively wrote Payment class first and then inherits from it; although we are not using its object in the system. Isn't it just increasing the code writing process; kindly explain?
Ans:
Purpose of inheritance is enhancing code reusability. Payment class is made parent class and cheque, charges and cash payment are child classes in given example. We have done it for reuse purpose. Later when you need any of these functionalities, the method of class Payment will be inherited by sub classes allowing us not to repeat those methods in sub classes again. In this way, it is not increasing code writing rather it is promoting code reuse and hence it will save our time.

what parameters are used to measure and analyze design quality?
Ans
here are four parameters used to analyze the quality of a software design. These parameters include efficiency, compactness, reusability, and maintainability.

Lec 31 :

In the switch statement, cases should always end with a break. An if statement is a useful alternative of switch statement.

Magic Numbers

These are constant that mean something but they do not give any indication of their importance or derivation, making the program hard to understand and modify. To a reader they work like magic and hence are called magic numbers.

It is much better to use symbols to explicitly indicate the intent of the statement.

Code example of High Coupling
Ans
Consider a simple shopping cart application that uses a CartContents class to keep track of the items in the shopping cart and an Order class for processing a purchase. The Order needs to determine the total value of the contents in the cart, it might do that like so:

```
public class CartEntry
{
   public float Price;
   public int Quantity;
}

public class CartContents
{
   public CartEntry[] items;
}

public class Order
{
   private CartContents cart;
   private float salesTax;

   public Order(CartContents cart, float salesTax)
   {
      this.cart = cart;
      this.salesTax = salesTax;
   }
```

```
    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += cart.items[i].Price * cart.items[i].Quantity;
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```

Notice how the OrderTotal method (and thus the Order class) depends on the implementation details of the CartContents and the CartEntry classes. If we were to try to change this logic to allow for discounts, we'd likely have to change all 3 classes. Also, if we change to using a List collection to keep track of the items we'd have to change the Order class as well. So the above examples represents the classis example of tight coupling.

Lec 32:
Abstraction and encapsulation are two important tools that can help in managing and mastering the complexity of a program.

The logical and operator, &&, and logical or operators, ||, are special due to the C/C++ short circuiting rule, i.e. a || b and a && b are short circuit evaluated. That is, logical expressions are evaluated left to right and evaluation stops as soon as the final truth value can be determined.

```
struct Node {
int data;
Node * next;
};
Node *ptr;
...
while (ptr->data < myData && ptr != NULL){
// do something here
}
```

What's wrong with this code?
ptr != NULL, is supposed to be the guard. That is, if the value of the pointer is NULL, then the control should not enter the body of the while loop otherwise, it should check whether ptr->data < myData or not and then proceed accordingly.

This would be the right choice !

```
while (ptr != NULL && ptr->data < myData){
// do something here
}
```

A side effect of a function occurs when the function, besides returning a value, changes either one of its parameters or a variable declared outside the function but is accessible to it.

As an example,
consider the following statement:

```
c = f1(a) + f2(b);
```

The question is, which function (f1 or f2) will be evaluated first as the C/C++ language does not specify the evaluation order and the implementer (compiler writer) is free to choose one order or the other. The question is: does it matter?
To understand this, let's look at the definition of f1 and f2.

```
int f1(int &x)
{
x = x * 2;
return x + 1;
}
int f2(int &y)
{
y = y / 2;
return x - 1;
}
```

In this case both f1 and f2 have side effects as they both are doing two things – changing the value of the parameter and changing the value at the caller side. Now if we have the following code segment,

```
a = 3;
b = 4;
c = f1(a) + f2(b);
```

then the value of a, b, and c would be as follows:

```
a = 6
b = 2
c = 8
```

So far there seem to be any problem. But let us now consider the following statement:

```
c = f1(a) + f2(a);
```

What will be the value of a and c after this statement?
If f1 is evaluated before f2 then we have the following values:

```
a = 3
b = 9 // 7 + 2
```

On the other hand, if f2 is evaluated before f2 then, we get totally different results.
a = 2
b = 3 // 3 + 0

how the objects are identified in peter codd's technique?
Ans
Peter Coad methodology divides the objects into different categories to make it easier to find them and establish their attributes, services, and collaborations. This activity, consisting of 6 steps is well explained in lecture 16. I recommend you to watch video lecture 16 as well as conslut same lecture from handouts for more details.

How can we decrease coupling?
Ans
Software coupling is defined as the degree to which a software module relies or depends on other modules. High coupling means that related classes have to know internal details of each other, changes ripple through the system, and the system is potentially harder to understand. Coupling can be reduced if we focus the following steps:
1. Make the code easier to read.
2. Make classes easier to consume by other developers by hiding the inner workings of classes behind well-designed APIs.
3. Isolate potential changes to a small area of code.
4. Reuse classes in completely new contexts.
This query has aklready been replied in lesson 14 mdb. I recommend you to consult mdb of every lecture. It will not only enhance your knowledge but also restricts you to ask repeated questions.

Lec 33:

Performance :

In many cases, performance and maintainability are at odds with one another. When planning for performance, one should always remember the 80/20 rule - you spend 80 percent of your time in 20 percent of the code. That is, we should not try to optimize everything.

When a small set (a couple of functions) of functions which use each other is so overwhelmingly the bottleneck, there are two alternatives:

1. use a better algorithm
2. rewrite the whole set

for (j = i; j < MAX_FIELD; j++)
clear(j);

This loop clears field before each new input is read. It was observed that it was taking almost 50% of the total time.

code was subsequently modified as shown below:
for (j = i; j < maxField; j++)
clear(j);
This reduced the overall execution time by half.

IS it necessary to show the connection between both database servers or both database servers are independent. Thanks
Ans
Its up to you. You can show connection between both data servers. Even if you don't show connect between them. There is no issue.

DIFFRENT PHARASE OF TESTING AND DIAGRAM?
Ans
Here ar different phases.
Unit testing : For individual components independent of other components.
Module testing : It is for testing a collection of dependent components
Subsystem testing: testing of collection of modules to discover interfacing problems among interacting modules.
System testing: It is for after integrating subsystems into a system
Acceptance test: validation against user expectations.
Alpha testing: testing for customized projects, in-house testing for products.
Beta testing: field testing of product with potential customers who agree to use it and report problem before system is released for general use

Lec 34 :

If we are careful during writing code, we can make it portable. On the other hand if we write code without portability in mind, we may end-up with a code that is extremely hard to port to other environment.
Guidelines to achieve Portability :

1. Use ANSI/ISO standard C++

2. Instead of using vendor specific language extensions, use STL as much as possible Sizes of data types cause major portability issues as they vary from one machine to the other so one should be careful with them.

Order of evaluation varies from one implementation to other.
The language does not specify whether char is signed or unsigned.

```
char c;
// between 0 and 255 if unsigned
// -128 to 127 if signed
c = getchar();
if (c == EOF) ??
// will fail if it is unsigned
```

It should therefore be written as follows:

```
int c;
c = getchar();
if (c == EOF)
```

Bit Fields :
Bit fields allow the packing of data in a structure. This is especially useful when memory or data storage is at a premium.

However, bit fields do suffer from a lack of portability between platforms:
 integers may be signed or unsigned
 Many compilers limit the maximum number of bits in the bit field to the size of an integer which may be either 16-bit or 32-bit varieties.
 Some bit field members are stored left to right others are stored right to left in memory.
 If bit fields too large, next bit field may be stored consecutively in memory (overlapping the boundary between memory locations) or in the next word of memory. Bit fields therefore should not be used.

WHAT ARE STATIC ANALYSIS AND HOW WE CAN CHEACK THE CHECK LIST?

MULTIPLE EXPRESSSIONS EXPLAIN?

Ans

Analysis of computer software performed without actually executing, or running, that software. Looking at applications in a non-runtime environment.

CHECK LIST:
Data faults, Control faults,Input/Output faults, Storage Management fault are cheked in static analyzer application.

DIFFRENCE BETWEEN ARCHITUCTURE AND SYSTEM'S?
Ans
Modularity is a tool that can help us in reducing the size of individual functions, making them more readable.
It increases the extent to which software is composed of separate, interchangeable components called modules by breaking down program functions into modules, each of which accomplishes one function and contains everything necessary to accomplish this.

The architecture of a system is its 'skeleton'. It's the highest level of abstraction of a system. What kind of data storage is present, how do modules interact with eachother, what recovery systems are in place.
While
System architectures hold the key to success or failure of a system every bit as much as the software architecture does for the software.

Lec 35:

Exception handling is a powerful technique that separates error-handling code from normal code. It also provides a consistent error handling mechanism.

One of the most powerful features of exception handling is that an error can be thrown over function boundaries. This allows programmers to put the error handling code in one place, such as the *main*-function of your program.

general exception handling mechanism :
try {
___...
___...
___throw Exception()
___...
___...

```
} catch( Exception e )
{
___...
___...
}
```

A number of invisible execution paths can exist in simple code in a language that allows exceptions. The complexity of a program may increase significantly if there are exceptional paths in it.

Providing the strong exception-safety guarantee often requires you to trade-off performance.
If a function has multiple un-related side-effects, it cannot always be made strongly exception safe.
Not all functions need to be strongly exception-safe.

1.LL ME GENRAL NAMING CONVENTION FOR C++ AND JAVA?

2.WHAT ARE CODE STRUCTURES?

Ans

Code structures:A program is usually not limited to a linear sequence of instructions. During its process it may bifurcate, repeat code or take decisions. For that purpose, C++ provides control structures that serve to specify what has to be done by our program, when and under which circumstances.For structural testing it is important to know about basic coding structures. There are four basic coding structures sequence, if statement, case statement, and while loop. These four basic structures can be used to express any type of code.
Plz consult handouts for naming conventions as these are well explained there.

1. .....WHILE LOOP SHOULD BE AVOIDED?

2.VE EQUIVALENCE PARTITIONING GUIDELINES?

3.WHAT ARE SYMPTOMS AND GIVE AN EXAMPLE OF MEMORY OVERRUN BUG CLASS?

Ans

1-do-while loops are less readable than ordinary while loops and for loops since the conditional is at the bottom of the loop. The reader must scan the entire loop in order to understand the scope of the loop. In addition, do-while loops are not needed. Any do-while loop can easily be rewritten into a while loop or a for loop. Reducing the number of constructs used enhance readbility.

2-Equivalence partitioning guidelines:
I-Organize your equivalence classes. Write them in some order, use some template, sequence, or group them based on their similarities or distinctions. These partitions can be hierarchical or organized in any other manner.
II_Boundary conditions: determine boundary conditions. For example, adding in an empty linked list, adding after the last element, adding before the first element, etc.
III- You should not forget invalid inputs that a user can give to a system. For example, widgets on a GUI, numeric instead of alphabets, etc.

3-Memory overrun occurs when you use memory that does not belong to you. This can be caused by overstepping an array boundary or by copying a string that is too big for the block of memory it is defined to hold. Memory overruns were once extremely common in the programming world because of the inability to tell what the actual size of something really was.

Lec 36 : Software Testing -> Software verification and validation :

processes in which we check a product against its specifications and the expectations of the users who will be using it.
Defect : A defect is a variance from a desired product attribute. These attributes may involve system specifications well as user expectation.
Software and Defect : Software and defects go side by side in the software development life cycle.
Software Testing : the process of examining the software product against its requirements. Thus it is a process that involves verification of product with respect to its written requirements and conformance of requirements with user needs.
Software testing objective : The correct approach to testing a scientific theory is not to try to verify it, but to seek to refute the theory. That is to prove that it has errors.
Testing limitations :

 In order to prove that a formula or hypothesis is incorrect all you have to do toshow only one example in which you prove that the formula or theorem is not working.
 On the other hand, million of examples can be developed to support the hypothesis but this will not prove that it is correct.

Test Cases and Test Data :

Thus a test case involves

 Input and output specification plus a statement of the function under test.
 Steps to perform the function
 Expected results that the software application produces

1,UNIT TESTING , SOFTWARE TESTING?

2,WHERE TERM COMPUTE CAN BE USED IN METHODS?

3, NON FUNCTIONAL REQUIREMENTS OF AN E-LEARNINGSYSTEM?

(A)USER FRIENDLESS,(B)TIME TAKEN TO DOWNLOAD (C)ONLINE ASSIGNMENT SUBMISSION FACILITY?

(DLINE CHATTING FACILITY?(E)REBUSTESS

Ans

1) Unit testing means, testing a single module. Normally this testing is performed by developer itself.

2) Please refer lecture no. 28 for this question.

3) There may be different non functional requirements e.g. availability, usability, security etc.

Please explain the Pattern and where and how we use it in software ? with examples regards Aorangzeb

Ans

In software development, there are certain problems (patterns) that are repeated again and again. The generic solution to these problems are called design pattern. E.g. for all the dynamic websites, the same style is followed i.e. there is a central server, which stores all the data, and clients access that data through browser. All the websites follow the same rule. So, we can say that the style (client/server) is a design pattern, as it is a common solution to a common occurring problem.

PLEASE TELL THAT SHOULD WE REDESIGN THE MODEL BY USING THE Façade Pattern OR IMPLEMENT IT IN THE C++….
Thanks.
Ans
You only have to re-design the model by implementing facade pattern. No need to provide any code.

Could you please explain little bit the main difference between Singleton Pattern & Observer Pattern.
Ans
In observer pattern, all the objects (views) continually synchronize with the central data object, i.e. whenever the data is modified all the views are also modified immediately (views in this way, always represent the latest data).

On the other hand the nature of singleton pattern is quite different, its implementation allows the instantiation of only a single object of any class on which singleton pattern is applied. I.e. you can not create more than one objects of this class.

Lec 37 : Testing vs. development :
 Functional specification document is the starting point, base document for both testing and the development
 Right side boxes describe the development, whereas, left side boxes explain the testing process
 Development team is involved into the analysis, design and coding activities.
 Whereas, testing team too is busy in analysis of requirements, for test planning, test cases and test data generation.

 System comes into testing after development is completed.
 Test cases are executed with test data and actual results (application behavior) are compared with the expected results,
 Upon discovering defects, tester generates the bug report and sends it to the development team for fixing.
 Development team runs the scenario as described in the bug report and try to reproduce the defect.
 If the defect is reproduced in the development environment, the development team identifies the root cause, fixes it and sends the patch to the testing team along with a bug resolution report.
 Testing team incorporates the fix (checking in), runs the same test case/scenario again and verifies the fix.
 If problem does not appear again testing team closes down the defect, otherwise, it is reported again.

testing phases : Unit testing , Module testing , Subsystem testing, System testing, Acceptance test, Alpha testing , Beta testing

Black box testing : not concerned with how the inputs are transformed into outputs. If the outputs match with the expected results, system is fine otherwise a defect is found.

Structural testing (white box) : we look inside the system and evaluate what it consists of and how is it implemented.

plz tell me about asynchronous error and synchronous error?

and my 2nd question is what is throwing error?

Ans

There are some statements in program that always throw error i.e. the error is raised in these statements at every execution of program, e.g the statement.

int x = y / 0 ;

will always generate an exception. Such errors are called synchronous errors.

On the other hand, some errors may occur at anytime, May occur in one execution but not in other, e.g. at any stage the program could not get the sufficient memory (may be due to number of other programs are executing). Such errors that are not specific to a statement but can occur at anytime are called asynchronous errors, these errors are very difficult to handle, and Exception handling helps us catching these errors.

Throwing error means raising an error and delegating someone, the responsibility to handle that error.

Sorry so asking such basic question. Software engineering is based upon computer science. what is the difference between IT and Computer science.
Regards
Ans

Computer science is a discipline that involves the understanding and design of computers and computational processes. Computer Science is about learning and understanding the mathematical, scientific and engineering principles underlying every kind of computing system, from mobile phones and the internets, through systems that interpret natural language, to the supercomputers that forecast tomorrow's weather or simulate the effects of disease on the human heart.
Where, Information technology is the study, design, development, implementation, support or management of computer based information systems, particularly software applications and computer hardware. IT deals with the use of electronic computers and computer software to convert, store, protect, process, transmit and securely retrieve information.
Basic difference between these two disciplines is that computer science focuses on computers and its processing where IT is concerned with the use of technology for information.

Lec 38: Equivalence Classes or Equivalence Partitioning : Two tests are considered to be equivalent if it is believed that:
 if one discovers a defect, the other probably will too, and
 if one does not discover a defect, the other probably won't either.
Equivalence partitioning guidelines :
Organize your equivalence classes. Write them in some order, template etc.
Boundary conditions: determine boundary conditions. For example, adding in/after an empty linked list etc
don't forget invalid inputs that a user can give to a system.
String matching
Organization : we divide the problem in two obvious categories of equal
strings and the other one for unequal strings. Within these equivalent partitions, further partitioning is done. You can see the examples in the handouts page 200.
Basis Code Structures : Flow graph notation :
we do not use flow graphs to describe decisions. That is, how a branch is taken is not shown in flow graphs.
I'd sugget you to have a look at flow graphs in the handouts for further clarification.

How can I use profiler in my programs; I use both Dev C++ am MS Visual Studio environment to write and compile my C++ programs; Does any of them have a built-in profiler? if yes, how can I use to check statistics of my code and optimize my programs?
Ans
Yes Dev C++ has a built-in profiler. To use the profiler in Dev C++, first you need to enable the profiling info generation option in Compiler Options. To enable this option, follow the steps given below:

Tools -> Compiler Options -> Code Generation/Optimization -> Generate profiling information for analysis

Set the Generate profiling information for analysis option to YES

After that follow these steps:

Tools -> Compiler Options -> Linker -> Generate debugging information

Set the option Generate debugging information to YES

Now next step is to open your program and then follow these options and rebuild and run your application:

Execute -> Rebuild All
Execute -> Run

After this, follow the steps below to view the profiling statistics

Execute -> Profile analysis

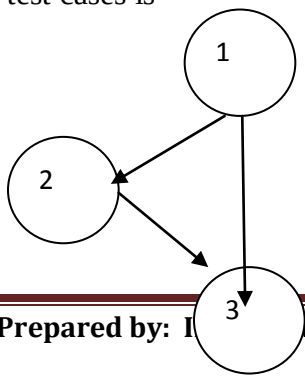
Reference to lecture no. 33, I did not get the difference between Row Major and Column Major. Can you explain in detail with example...
Ans
Row major and Column major are data organizational techniques used in programming languages. More, specifically, these are methods for for storing multidimensional arrays in linear memory. In row-major storage, a multidimensional array in linear memory is accessed such that rows are stored one after the other. Where in Column major storage, a multidimensional array in linear memory is accessed such that columns are stored one after the other. Row major is used in C where Column major is technique of scientific languages like FORTRAN.

what is software and test ?

Ans

Software is a program/collection of programs that are used to operate computer. Test is examination of software to make sure that it is performing required functionality.

Lec 39 :
White box testing : Coverage :
 Statement Coverage: In this scheme, statements of the code are tested for a successful test that checks all the statements lying on the path of a successful scenario.
 Branch Coverage: In this scheme, all the possible branches of decision structures are tested. Therefore, sequences of statements following a decision are tested.
 Path Coverage: In path coverage, all possible paths of a program from input instruction to the output instruction are tested. An exhaustive list of test cases is generated and tested against the code.
Example :
if (a = = b) //1
c = d; //2
a++; //3
Statement coverage:
 a=1,b=1



 If a==b then statement 2, statement 3
Branch coverage
 Statement 1: two braches 1-2, 1-3
 Test case 1: if a =1, b=1 then statement 2
 Test case 2: if a=1,b=2: then statement 3
Path coverage
Same as branch testing
Cyclomatic complexity : It defines number of independent paths in the basis set of a program.
Cyclomatic Complexity, V(G), for a flow graph G is defined as: V(G) = E - N + 2
Where E is the number of edges and N is the number of nodes in the flow graph G.

Bit fields allow the packing of data in a structure. This is especially useful when memory or data storage is at a premium

I read this line in handouts. Kindly explain what does premium mean in the above sentence? I mean what is the sense of this word in this sentence. Does premium mean that memory can have chances to be leaked?

Ans

Packing the data in bit fields means you use individual bits in variables to store the data rather than using separate variables. So, bit fields let you save your memory e.g. suppose you have a system which maintains the status of eight different hardware appliances, and you use a character variable to store the status of each appliance. Then storing the status of eight appliances means 8 character variables (each of 8 bits and thus total 64 bits consumption).

Now suppose you use a single character variable of 8 bits and use each of its bits to store the status of each appliance, this means that you need only one variable (of 8 bits) and not eight separate variables, so you have saved the memory of 7 variables.

Thus bit fields let you save your memory. This is very important when memory is very expensive or you are paying some amount to use memory. So, saving memory in this way means, saving the money. Premium means when memory is purchased and using memory is an expensive task.

what is meant by Hungarian Notation:

Shall I write Medicine NAme as medicineName or MedicineName. Please kindly explain

Ans

Hungarian notation is a naming convention in computer programming, in which the name of a variable indicates its type or intended use.

Lec 40 : Unit testing : A software program is made up of units that include procedures, functions, classes etc. The unit testing process involves the developer in testing of these units.
Software should be tested more like hardware, with
 Built-in self testing: such that each unit can be tested independently
 Internal diagnostics: diagnostics for program units should be defined.
 Test harness
Unit Testing Principles
 In unit testing, developers test their own code units (modules, classes, etc.) during implementation.
 Normal and boundary inputs against expected results are tested.

 Thus unit testing is a great way to test an API (*Application programming interface*)

Unit Testing Tips :
 For small projects you can imbed the unit test for a module in the module itself
 For larger projects you should keep the tests in the package directory or a /test subdirectory of the package. By making the code accessible to developers you provide them with:
 Examples of how to use all the functionality of your module
 A means to build regression tests to validate any future changes to the code
Inspection and chaotic zone : if defects are not discovered and fixed at the appropriate stage, then at the testing and maintenance phases, these defects are piled up becoming chaotic zone for the testing and developing teams.

Please explain to me the exception handling. what is this throwing and catching. Explain the whole construct to me thanks

Ans

Exception Handling
Exception handling is a programming language construct or computer hardware mechanism designed to handle the occurrence of a condition that changes the normal flow of execution. For signaling conditions that are part of the normal flow of execution, see the concepts of signal and event handler. In general, the current state will be saved in a predefined location and the execution will switch to a predefined handler. Depending on the situation, the handler may later resume the execution at the original location, using the saved information to restore the original state. For example, an exception that will usually be resumed is a page fault, while a division by zero usually cannot be resolved transparently. From the processing point of view, hardware interrupts are similar to resumable exceptions, although they are usually not related to the current program flow. From the point of view of the author of a routine, raising an exception is a useful way to signal that the routine could not execute normally. For example, when an input argument is invalid (a zero denominator in division) or when a resource it relies on is unavailable (like a missing file, or a hard disk error). In systems without exceptions, routines would need to return some special error code. However, this is sometimes complicated by the semipredicate problem, in which users of the routine need to write extra code to distinguish normal return values from erroneous ones. In runtime engine environments such as Java or .NET, there exist tools that attach to the runtime engine and every time that an exception of interest occurs, they record debugging information that existed in memory at the time the exception was thrown (call stack and heap values). These tools are called Automated Exception Handling or Error Interception tools and provide 'root-cause' information for exceptions.

i wnat to become a software engineer so what courses i should specialize an kind of studey i will do

Ans

For the said purpose you need master the one specific field in Software Engineering. There are different areas in Software Field but you may choose one of the below.:-
Software Developer
Software Quality Assurance Engineer

Software Process Engineer
System Analyst
It depends upon you in which area you are interested in.

plz define that waht are classes and interfaces

Ans

objects define their interaction with the outside world through the methods that they expose. Methods form the object's interface with the outside world; the buttons on the front of your television set, for example, are the interface between you and the electrical wiring on the other side of its plastic casing. You press the "power" button to turn the television on and off. In its most common form, an interface is a group of related methods with empty bodies.
Now we talk about Classes
Classes
In object-oriented terms, we say that your bicycle is an instance of the class of objects known as bicycles. A class is the blueprint from which individual objects are created.

Lec 41:
Inspection versus Testing : Inspections and testing are complementary and not opposing verification techniques.
Both should be used during the verification and validation process.

Inspection checklists.
Checklist of common errors in a program should be developed and used to drive theinspection process. These error checklists are programming language dependent such thatthe inspector has to analyze major constructs of the programming language and develop checklists to verify code that is written using these checklists.

Static analyzers
Static analyzers are software tools for source text processing. They parse the program text and try to discover potentially erroneous conditions and bring these to the attention of the verification and validation team.

i want to know that keyword are the part of Coding Styles / Conventions or not. can i find differences of java and C# coding style on the basis of keywords.

Ans

Yes may find differences in this aspect. But you need to foucs on other areas as well.

what is meant by multiple increments?
Ans
multilple increments means" deliver multiple parts of the software system with the passage of time and with increment(additional features/functionality) to the customer.

Lec 42:
Importance of Debugging :
According to a survey,when a software application is in the maintenance phase, 20% of its lifecycle cost is attributed towards the defects which are found in the software application after installation. Please bear in mind that the maintenance is the phase in which 2/3rd of the overall software cost incurs.

Therefore, 20% of the 2/3rd cost is again a huge cost and we need to understand why this much cost and effort is incurred. In fact, when a software application is installed and being used, any peculiarity in it can cost a lot of direct and indirect damages to the organization.

and Science of Debugging:
Debugging is taken as an art but in fact it is a scientific process. As people learn about different defect types and come across situations in which they have to debug the code, they develop certain heuristics. Next time they come across a similar situation, they apply those heuristics and solve the problem in lesser time and with a lesser effort.

1) Black Whole, when data is not processed ?
2) Miracle, There is no any input but there is some out put
What we mean by synch ?
what we mean by gray whole?

Ans

These are processes containing some mistakes (DFD common mistakes):
Black hole: A process which takes input but does not generate any output.
Miracle: A process which does not take any input but produces an output.
Gray hole: A process in which output is produced without required and suffieicent inputs.

what does we mean by
1) #ifndef
2) #define
3) #endif
4) %
I saw such statements in many C codes, % is used many times with printf ()

Ans

These are preprocessing directives used as different conditional parameters to control the compiling of source code.

From lecture 37 i have come to conclusion that structural testing is more elaborate and better as compared to black box....is it correct?
Ans
We can not comment that structural testing is more better than black box. There are two things; in large and complex embedded system, if we want to identify which module is problematic, we will do black box testing as only white box testing will be time consuming. Secondly, in complex and simple both systems, first black box would be used to identify the module/system containing issue and then structural testing would be used for further analysis. So both types are equally effective. And testing is incomplete if any one of them is not done.

Lec 43:
A memory leak bug is one in which memory is somehow allocated from either the operating system or an internal memory "pool", but never deallocated when the memory is finished being used.

Symptoms

- System slowdowns
- Crashes that occur "randomly" over a long period of time

Logical Errors

A logical error occurs when the code is syntactically correct but does not do what you expect it to do.

Symptoms

- The code is misbehaving in a way that isn't easily explained.
- The program doesn't crash, but the flow of the program takes odd branches through the code.
- Results are the opposite of what is expected.
- Output looks strange, but has no obvious symptoms of corruption.

Coding errors: A coding error is a simple problem in writing the code.

Symptoms

- Unexpected errors in black box testing.
- The errors that unexpectedly occur are usually caused by coding errors.
- Compiler warnings.
- Coding errors are usually caused by lack of attention to details.

Memory over-runs: a memory overrun occurs when you use memory that does not belong to you. This can be caused by overstepping an array boundary or by copying a string that is too big for the block of memory it is defined to hold.

Symptoms

- Program crashes quite regularly after a given routine is called, that routine should be examined for a possible overrun condition.
- If the routine in question does not appear to have any such problem the most likely cause is that another routine, called in the prior sequence, has already trashed variables or memory blocks.
- Checking the trace log of the called routines leading up to one with the problem will often show up the error.

Loop Errors

- Loop errors break down into several different subtypes.
- They occur around a loop construct in a program.
- Infinite loops, off-by-one loops, and improperly exited loops.

Symptoms

 If your program simply locks up, repeatedly displays the same data over and over, or infinitely displays the same message box, you should immediately suspect an infinite loop error.

 Off-by-one loop errors are quite often seen in processes that perform calculations.

 If a hand calculation shows that the total or final sum is incorrect by the last data point, you can quickly surmise that an off-by-one loop error is to blame.

 Likewise, if you were using graphics software and saw all of the points on the screen, but the last two were unconnected, you would suspect an off-by-one error.

 Watching for a process that terminates unexpectedly when it should have continued.

Pointer errors:

 A pointer error is any case where something is being used as an indirect pointer to another item.

 Uninitialized pointers: These are pointers that are used to point at something, but we fail to ever assign them something to point at.

 Deleted pointers, which continue to be used.

 An Invalid pointer is something that is pointing to a valid block of memory, but that memory does not contain the data you expect it to.

Symptoms

 The program usually crashes or behaves in an unpredictable and baffling way.

 You will generally observe stack corruptions, failure to allocate memory, and odd changing of variable values.

 Changing a single line of code can change where the problem occurs.

 If the problem "goes away" when you place a print statement or new variable into the code that you suspect contains the problem.

Boolean bugs

Boolean bugs occur because the mathematical precision of Boolean algebra has virtually nothing to do with equivalent English words.

Symptoms

 When the program does exactly the opposite of what you expect it to. For example, you might have thought you needed to select only one entry from a list in order to proceed. Instead, the program will not continue until you select more than one. Worse, it keeps telling you to select only one value.
 For true/false problems, you will usually see some sort of debug output indicating an error in a function, only to see the calling function proceed as though the problem had not occurred.

Test cases are part of ATP?
Ans
Test cases corresponds to application functionality which are required to user. Where ATP is complete procedure of testing a system. Yes, test cases are part of ATP.

Whose responsibility is it to write the Acceptance Test Procedure (ATP)?
Ans
Acceptance test procedure is responsibility of quality assurance team/department

How do a test Engineer carry out unit testing of the c++ code. As we have learned in OOP a class e.g. is dependent upon so many outer variables and other objects as well, so how will the test engineer test it independently? Is he suppose to write another piece of code to test each class separately? ....is there any off the shelf tool available for this purpose?
Ans
Unit testing is a two way work as developer is doing unit testing by himself when he is writing code. When quality assurance team or test engineer performs test, they mention error containing module/block to developer for testing because test engineer can't do unit testing of code. It is done by the develper. When developer is asked then he again checks the code for issues. Code analysis, code checkers and code optimizers are also used for this purpose.

Please tell me about the term beta release or beta testing in simple words. Regards
Ans
In beta testing, some customers/users are provided with developed product. They use it and provide feedback about product which is (if required) incorporated in the ptoduct. After that, it is delivered for general use.

What is test cases and what is test data?
Ans
A test case is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. Where test data is data which has been specifically identified for use in tests. Some data may be used in a confirmatory way,

typically to verify that a given set of input to a given function produces some expected result. Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input.

what is the basic difference between test cases and test data? Please explain this with an example. I shall be thankful to you. Regards,
Ans

A test case is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. Where test data is data which has been specifically identified for use in tests. For example, you want to develop a test-case for login functionality of LMS. Set of conditions defined (Pre-condition, Step, Action, Expected Response, Pass/Fail, Comments, Post-conditions) is test case. When you give a user name and pasword in LMS to check login functionality then this user name and password is test data.

Lec 44:
Holistic approach:
Holistic means
 Emphasizing the importance of the whole and the interdependence of its parts.
 Concerned with wholes rather than analysis or separation into parts
The debugging process:
First hand accounts of the problem are always useful, so be sure to write
down exactly what you are told. That way, you can compare several accounts of the same problem and look for similarities.
Good clues, Easy Bugs
Get A Stack Trace
Following stack trace information may help in debugging process.
 Source line numbers in stack trace is the single, most useful piece of debugging information.
 After that, values of arguments are important
o Are the values improbable (zero, very large, negative, character strings

with non-alphabetic characters?
 Debuggers can be used to display values of local or global variables.
o These give additional information about what went wrong.
Non-reproducible bugs
 Bugs that won't "stand still" (almost random) are the most difficult to deal with.
Randomness itself, however, is information.
 Are all variables initialized? (random data in variables could affect output).
 Does bug disappeared when debugging code is inserted? Memory allocation (malloc) problems are probably a culprit.
 Is the crash site far away from anything that could be wrong?
 Check for dangling pointers.

What is the effective testing?
Ans
he objective of testing is to discover the maximum number of defects with a minimum number of resources before the system is delivered to the next stage. If this objective is achieved then it is effective testing. That is, testing is effective enough to ensure the product rightness.

what is the goal of the tester person?
Ans
Responsibility of the tester is to achieve testing objective which is, to discover the maximum number of defects with a minimum number of resources before the system is delivered to the next stage. A good tester does not rely on few times testing of the program but carries out a thorough analysis of the program to devise test cases that can be used to test the system systematically and effectively from every perspective of testing each and every functionality.

Q:Describe the importance of software debugging? Q: Differentiate Hungarian notation,Bi capitalization with example?
Ans
Both Hungarian and Bicapitalization are two naming conventions used to name the variables or identifiers in the code.

Hungarian Notation is mainly confined to Microsoft Windows programming environments, such as Microsoft C, C++ and Visual Basic. Hungarian notation involves storing information about the variable in the very name of the variable.

With Hungarian notation, variable names are separated into two parts:

1. The lowercase prefix, which contains information about the variable type, and
2. The qualifier, which tells you what the variable contains. The qualifier usually begins with a

capital letter.

For example: pstrError

BiCapitalization notation is a common name for the practice of writing compound words or phrases where the words are joined without spaces, and each word is capitalized within the compound.

For example: totalMarks
numOfStudents

How this profiling and performance tuning is affect on SE program?
Ans
It is hard to understand the behavior of the program statically therefore a thorough analysis based on runtime data is needed. A Performance profiling is a technique to investigate runtime behavior of the program. The purpose is to identify the bottlenecks in the program with the objective to increase optimization.

What we mean by profiler, is it some kind of software, please guide me.
Ans
Profilers are used in the performance engineering process. Profilers are software used to perform profiling. Profiling is a way to measure where a program spends time and helps to uncover performance problems in the code. Profilers use a wide variety of techniques to collect data, including hardware interrupts, code instrumentation, instruction set simulation, operating system hooks, and performance counters.

Is Unit Testing is white box testing or it is black box testing?
Ans
Unit testing is testing individual components independent of other components. It is performed to check an individual component. So it is both, black box testing as well as white box because we check component input/output along with complete program structure.