1. INTRODUCTION
The software is a trillion dollar business, and the amount of effort and time spent to develop software is very large. In time of software developing, different architectures are used for the development and security is a major concern in software components. Security issues should be resolved in the early phases of development, and costs incurred to remove the objection if detected in the architecture can be minimum [1]. If the software development is performed in coding phase, deep diagnosis is needed because finding the design faults in the code is difficult. Risk analysis should be done for the architectures because the software security plays a very important role in the program. Risk identification is useful and can be used to take the software security measures in to consideration. Determination of risk quantifies and its impact should be calculated in software development environment because it can have direct concern in this trade.

One question that organizations usually face is how my system is safe. The answer to this question is often difficult. Most of system securities problems are rooted in software which are most vulnerable to the unexpected attacks [2]. In spite of the extensive researches in security engineering, security measurement is still a difficult problem. Though we do not have the security measure with full confidence, we mostly rely on risk measure to assess the security. The use of security measures to assess the risks of aggression is a common practice which is provided by a systematic mechanism for cost and resource optimization. The problem is to provide accurate information on the attacks and their probability. Because the systems are subject to constant changes, the related risks are also affected by these changes. However, risk assessments are not repeated often enough to change the system. Over the time, the initial estimates of the risks got obsolete and maybe result in less secure systems.

2. SOFTWARE ARCHITECTURE
The software architecture is an essential for the proposed pyramid analysis method. Each of the components must be identified and their interactions with other components of the system should be considered in the DTM model. System software architecture should be available in the standard form. Information of the components interactions should be estimated using the experience gained from the similar software components. A standard method for estimating the probabilities is transition of the inter-components flow control from different scenarios run on the operations system profile, reliability, efficiency, demand, and quantitative information about their individual components. The software development will continue, the estimation will improve and thus throughout the analysis accuracy will improve.

3. TEST STRATEGY
Architecture provides the structure through which the large complex system can be understood. Before the system construction starts, weaknesses are identifiable. To create such a structure, system architecture selects a set of components and connections between them to display and thus create a specific architecture. The architecture of a software system is more complicated than that of a traditional system lifecycle. Certain areas of the code because of the information hiding techniques available in object-oriented languages are unavailable, and this may prevent some of the testing techniques. The object-oriented design information hiding techniques require different tests. Testing of object-oriented systems is more complicated than that of a traditional system lifecycle. Separate classes with integrity are designed to be compatible. It is possible to give specific access scores to a separate class of test without adjusting the design integrity. In the picture, different phases of the software development life cycle (SDL) are shown. If we let the requirements change in the next phases except for the software requirements, requirements change over time and the requirements lied in the previous phases may be invalid. In object-oriented software design, easy maintenance and reuse of components is the point focused on [4]. Software quality attributes such as integrity, strength, develop ability, and compatibility should also be considered during the design process. The main focus is the design phase of the test is on the test integrity. The test integrity test if systematic components construction is in the sub-systems and systems. The components variation...
sand whether the components properly transfer and control the data guarantees the successful integrity of the dependent components. For integrity test, the functional test of the box and the structural test or white box is frequently performed to separate the system components and determine the correct behavior of the integrated units. Drivers and specific tools are needed for separating the components and testing the individual components because they were integrated. Error reporting can be curtailed at a time of function requirement specification (SRS) to maintain even when the system is working. Object-oriented approach is modular in nature; therefore, it helps in easy maintenance and correcting the errors.

4. TESTING OBJECT-ORIENTED SYSTEM

Software testing techniques have been evolved over the years and for management of unique issues of object-oriented software testing, traditional software testing techniques must be adapted and new techniques have to come into existence. Object-oriented software architecture is different from traditional software architecture. Features such as hiding, inheritance, polymorphism, and reusability are specific for object-oriented software. Therefore, some of the issues related to object-oriented software testing are different from traditional software testing issues. The main advantage of the object-oriented paradigm is that it provides a uniform structure for all components of the interface process [5]. The focus class of the unit test is in object-oriented software. Resolving the problems at runtime complicates the testing for dynamic annexing. In addition, the paradigm shifts from waterfall model of software development to iterative and incremental model of software development, and this leads to the fact that the object-oriented software testing becomes repetitive and increasing in nature. The errors can appear anywhere in the code. Object-oriented class methodology helps to detect errors by providing check point from both type of interpretation (compile) and run-time errors for class objects. Check type of the runtime will detect large number of errors because the invalid objects (a lot of errors cause) are automatically detected and reported. For better system performance, the programming language should avoid leak memory. Leak memory logically is allocation of dynamic storage which leads the program to fail in the second request. Namely, the objects which are not used any more do not correct. On the contrary, a large number of samples may lead to leak memory. If the leak memory is severe, it may lead to failure of the program due to memory reduction. Leak memory is occurred by objects that help other objects to continue holding the references, accordingly, they prevent the garbage collecting to request the held in objects. Table of resource objects can be used to identify such resources.

5. OBJECT-ORIENTED SOFTWARE ARCHITECTURE

The object-oriented systems may be based on the following architectures:

5.1 Component Object Model (COM)

COM has Application Programming Interface (API) medium which support component creation for use in a particular application integration. COM has several components to interact during API design. Nevertheless, to interact the components should resort to a binary structure specified by Microsoft. Until the components resort to this binary structure, the written components can work with different languages.

5.2 Distributed COM (DCOM)

DCOM is an extension of COM which causes the interaction between network-based components. While COM processes can be run on one machine but in different address spaces, DCOM allow the processes to expand on a network. As DCOM environment will be available, the components running on different platforms can interact with DCOM. To obtain a COM / DCOM architecture, component-level testing is important [6]. Three types of property should be included in the definition of the functional part of the component which the application state of the test can be made. Every operation, in terms of the restrictions, is defined on its input and output which have been shown in pre-conditions and post-conditions form.

5.3 Common Object Request Broker Architecture (COBRA)

Object management group creates the COBRA to make it possible for the objects to connect with each other and supports the layer server client from the distributed objects, and data processing need not extra processing on PC to complete the transaction. Object shaves platforms which are dominant in terms of space and sharing because they share the resources. All errors or faults should be tested before the component is integrated in the distributed system. It should be tested aside from others so that it can find and remove the system errors and fault. This test should be done with unit test and integration test. All the uncertainties and implementation of interface inconsistencies should be found before administration of medium identification.

5.4 Model-View-Controller (MVC)

The data model, user medium, and control logic have been separated into three distinct components in MVC; changes in one component can be done with minimal effect on other parts of the MVC.

6. SECURITY CONSIDERATIONS

Software security considerations appear from the first phase of the software development even before the emergence of the problem domain-specific architecture. Almost all models and known architectures such as COBRA, EJB, COM, and DCOM take the security as the primary consideration in software architecture. Because COM / DCOM components have access to API and Microsoft Windows, bad actors can damage the user's computing environment. To consider this problem, Microsoft applies authoritative code which uses common coding for digital signature of the components. Independent operator of certification issuance such as VeriSign are used for identity confirmation of the component source, but even the certified code may include some orders which can affect the user environment either by accident or maliciously.

7. SECURITY RISK ANALYSIS

We use the term risk analysis to refer to risks identified and ranking activities in a particular stage in the life cycle of software development. Risk analysis particularly when used in architectural and product design is popular. Most descriptions of risk analysis emphasize that identification, ranking, and risk reduction are continuous process and not a single step in the development stage which is completed in the life cycle. Risk analysis results and risk categories thus drive both into requirements (early in the lifecycle) and into testing (where risk results can be used to define and plan particular tests). An architectural risk analysis method contains several main activities which most include the following result:
Discussion
Learns much as possible about the objective analysis.

- Read and understand the definitions, architecture documents, and other redesigning issues.
- Play with the software (if there are executable).
- Get a discussion about the goal.
- Specify the boundaries between system and sensitivity/criticality of the dataset.
- Read the code and other software products (including code analysis tool).
- Identify threats and agree on the attack sources. Discuss about the security issues that

Surround the software
Talk about how the product uses and determine areas of disagreement or ambiguity.

- Identify the areas which are at possible risk, sometimes use at risk tools or lists.
- Identify the behavior and discuss about the possible corrections.
- Gain an understanding of current security controls and planning.

During the process of architectural risk analysis, we follow the basic steps. Risk management is the fractal theory. In other words, the whole proceeding process can be applied in several different levels. The main level is that of the project. Each stage of validation cycle may have a display with the requirements, design, architecture, and other things used for testing programs.

8. COST-BENEFIT ANALYSIS
Architecture selection and analysis is performed with cost benefit analysis and can help in the economic aspects of software molded to help determine the optimum. Methods of cost-benefit analysis (MCBA) help to determine suitability of software architecture for the question and answer application environment steps. CBAM has many steps, and each step is checked and then implemented in detail and thus justify the cost of software development. While selection of scenarios and strategies are considered for the systems security, architecture strategies should be chosen for a particular scenario. For example, if there is no additional risk to the system, then specialize the strategy should be chosen which should add to the system’s security and the ability to overcome failure. The quality assurance and quantity determination of architecture strategies extract the benefit information from commercial concepts and the system conditions (what is supposed to be better than all the commercial concepts the changes in the system and how it functions) to aid architecture strategy. Benefits better understand that a strategy reaches a favorable level of qualification feature than the architects. In the fourth stage, costs and quantify pick architectural strategy concepts are quantified. The costs are extracted and information is planned from shareholders. There is no special technique for this extraction. It is assumed that in the organization there is a cost estimation and plan method to achieve this goal. Based on the extracted values, the desirability degree is calculated for architecture. One measure of desirability degree (divided by the cost-benefit ratio) is performed for each architecture strategy. Inherited uncertainties in each of these values which can also help in the final stages of decision-making are calculated. The above six steps help to calculate the extracted amount as an acceptable basis for decision-making process which include technical measures and trade-criteria for determining whether particular change is the system provides a high return on investment and the suitability or not (ROI).

9. ARCHITECTURE SECURITY MODEL
Creating a component-level security and security architecture model of the software is not easy because there should be a tool to provide information for the applications and IT systems to extract, categorize and prioritize security requirements and should enter its security needs at the architecture level. This methodology focuses on security considerations entering in to the early stages of the development life cycle. The model may be useful for documenting and analyzing the security aspects of the respective systems and may be used to guide future developments and changes in these systems. This model is elaborate on the development of the security architecture.

10. CONCLUSION
Security issues are discussed in detail. All software architectures used for software development are discussed in this article. Security is essentially necessary for software architecture because remove the loopholes in the system and it can be done with risk analysis. In this method, commercial objectives determine the risks, the risks guide the methods, the methods gain the measurement, the measurement leads the decision support, and decision support directs the back work and program quality. We used the usual notions of security engineering to create a model to assess security. The model was risk-based, which is widely accepted as a measure of security. Our approach emphasizes that each part of the system own its risk, no matter how small it is. Software components, people, and communication channels provide security risks. Only one comprehensive analysis of all of these provides real results on the system security. Requirements of a system should be finalized in the design phase.

REFERENCE