Security Concerns of Object Oriented Software Architectures

Dinesh Kumar Saini
Associate Professor, Faculty of Computing and IT, Sohar University, Oman
Research Fellow, Faculty of Engineering, Information Technology and Architecture
University of Queensland, Brisbane, Australia

ABSTRACT
Testing and measuring the security of software system architectures is a difficult task. An attempt is made in this paper to analyze the issues of architecture security of object-oriented software's using common security concepts to evaluate the security of a system under design. Object oriented systems are based on various architectures like COM, DCOM, CORBA, MVC and Broker. In object oriented technology the basic system component is an object. Individual system component is posing it own risk in the system. Security policies and the associated risk in these software architectures can be calculated for the individual component. Overall risk can be calculated based on the context and risk factors in the architecture. Small risk factors get accumulated together and form a major risk in the systems and can damage the systems.

General Terms
Software, Object Orientation, Architecture, Software Engineering,

Keywords— Security, COM, DCOM, CORBA, Test Strategy, risk, SDLC

1. INTRODUCTION
Software is a trillion dollar business and the amount of effort and time spend for developing software is huge [1]. While developing software different architectures are used for development and security is one of the main concerns in the software components [2]. Security related issues should be resolved in the early phases of the development and the expenses incurred for removing defects is minimal if it get detected at the architectural level [3]. Higher level of understanding is required if the software development is in coding phase and design defects are not easy to detect in the code [4]. Risk analysis must be carried out for architectures because it plays very important role in software security program [5,9]. Risk identification is very helpful and software security measures can be taken with help of it. Risk quantification and its impact must be calculated in the software development environment because it has direct concerns in the business. System level concerns with probability and impact measures must be handled carefully while developing the software. One of the major questions organizations often face is "how secure are my systems?" Answering such a question is often difficult. The root of most security problems is software that fails in unexpected ways when under attack [7, 10]. Despite extensive research in security engineering, measuring security is still a difficult problem [12, 13, and 14]. While we do not have security measurements with absolute certainty, we often rely on measurement of risk in assessing security. Using risk of violations to evaluate security decisions is a common practice. It provides a systematic mechanism for optimizing cost and resources. The difficult part lies in providing accurate information on attacks and their likelihood [25, 29]. Since systems are typically exposed to constant changes, associated risks are often affected by such changes. However, risk assessments are not typically repeated as often as changes are introduced into systems. Over time, initial risk estimates become outdated possibly leading to less secure systems.

2. SOFTWARE ARCHITECTUR
Software architecture knowledge is essential for applying the proposed hierarchical analysis Approach. The individual components need to be identified and their interaction with other components needs to be taken into account for constructing the DTMC model corresponding to the system. Software systems architecture is to be made available in the standard form [30, 50].The information regarding interaction among components must be estimated with experience gained with similar software components. A standard method to estimate the control flow transition probabilities among components from the occurrence probabilities of various execution scenarios based on the operational profile of the system. Reliability, performance and demands, etc., are the quantitative information regarding the individual characteristics of the components [32]. As the software development continues, the estimates become better and the analysis thus improves with time in terms of accuracy.

3. TEST STRATEGY

Architecture provides a structure through which a large or complex system can be understood and reasoned about. Weaknesses can be identified before the system is built. In creating this structure, the system architect chooses to represent a set of components and various connections between the components while abstracting away other details of the system, thus forming a particular perspective [1]. The architecture of a software system defines that system in terms of components and of interactions among those components [6]. Different choices of components or connections will provide different architectures, and therefore different perspectives, of the system. Testing must be an integral part of the complete software development cycle [3]. The exact way of the testing process is dependent on the type of the software development life cycle. The cycle may be incremental or iterative and accordingly the testing software should be developed using the same techniques as the production software. Object-oriented design techniques of encapsulation and information hiding require different testing techniques. Testing for object-oriented software system is more difficult and more complex than for traditional system

life cycle [44, 45]. Certain areas of the code are "unreachable" due to the information hiding techniques available in object-oriented languages. This may hinder usual testing techniques. A dedicated test class can be given special access privileges without compromising the integrity of the design. In fig.1 the different phases of Software Development Life Cycle (SDLC) are demonstrated. If we allow requirements to be changed in later phases other than Software Requirement Specification (SRS) it will means that the needs of the users of a software system may change over time, invalidating the requirements laid down in an earlier phase. In object-oriented software design the emphasis is on easy maintenance and reuse of the components [10]. Software quality attributes like correctness, robustness, extensibility, and compatibility must also be addressed during design.

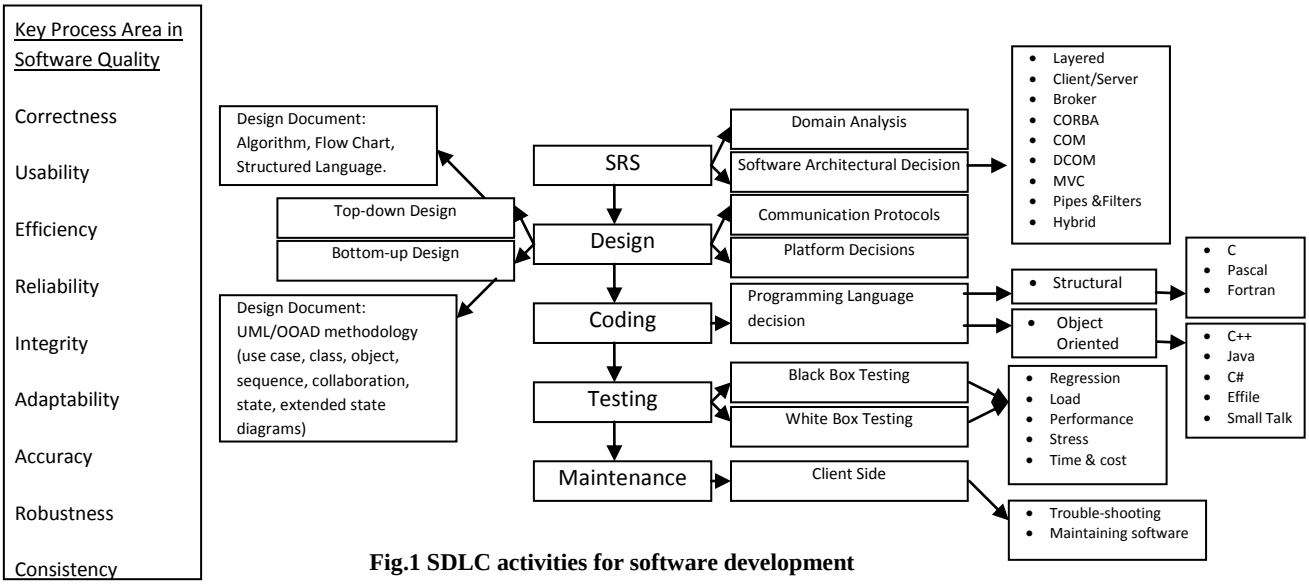

Fig.1 SDLC activities for software development

The primary focus of testing at the design phase is integration testing. Integration testing is the systematic composition of components into sub-systems and systems [11]. Tests ensure the consistency of component interfaces and whether the components pass data and control correctly, which results in successful integration of dependent components [44]. Integration testing requires that functional or black box testing and structural or white box testing be applied intermittently as necessary to isolate component functions and determine correct behavior of integrated units [45]. Specialized testing tools test drivers and stubs are needed to isolate components and test separate components as they are integrated. Figure 1 describes all the stages of object oriented software stating from SRS to maintenance. Key process areas of software quality with software architectures and design mythologies are described in the figure1. Error or bug can appear any time from SRS to maintenance even when the system is in the operation. Object oriented approach is modular in nature so it helps in easy maintenance and correction of bugs.

4. TESTING OBJECT ORIENTED SYSTESM

Software testing techniques have evolved over the years and in order to handle the unique testing issues of the object oriented software, the conventional software testing techniques need to be adapted and new ones need to be developed [26,27]. The architecture of the object-oriented software differs from the conventional software architecture. Features like encapsulation, inheritance, polymorphism and reusability are unique to object oriented software. Thereby, some issues involved in the testing of the object-oriented software are different from the testing issues of the conventional software. The key advantage of the object-oriented paradigm is that it provides a uniform structure for all components in the form of a procedural interface [12]. Class is the focus of unit testing in object-oriented software. Resolution of relationships at runtime due to dynamic binding complicates testing. Moreover, the paradigm shift from the waterfall model of software development to the iterative and incremental style of software development has resulted in object-oriented software testing being iterative and incremental in nature. A bug may appear anywhere in the code. The object oriented class methodology helps to detect bugs by providing for both compile-time and run-time type checking of pointers (handles) to class objects. This run-time type checking catches a lot of bugs since invalid object handles (the cause of a lot of bugs) are automatically detected and reported. For better performance of the system the programming language must avoid the memory leakage problem [43, 49]. A memory leak is an error in a program's dynamic store allocation logic that causes it to fail to reclaim discarded memory. That is, objects that are no longer required are not reclaimed. Unexpectedly large numbers of such instances may suggest a memory leak. A memory leak, if severe, can lead to the collapse of the application due to its running out of memory [22, 23]. Memory leaks are caused by objects that continue to hold references to other objects, thus preventing garbage collection from reclaiming the held objects. The Object References table can be used to help identify such references.

5. OBJECT ORIENTED SOFTWARE ARCHITECTURE

Object oriented systems may be based on following architectures:

5.1 Component Object Model (COM)

COM has application-programming interface (API) which supports for the creation of components for use in integrating custom applications. COM has diverse components to interact while designing the API. However, in order to interact,

components must adhere to a binary structure specified by Microsoft. As long as components adhere to this binary structure, components written in different languages can interoperate.

5.2. Distributed COM (DCOM)

DCOM is an extension to COM that allows network-based component interaction. While COM processes can run on the same machine but in different address spaces, the DCOM extension allows processes to be spread across a network. With DCOM, components operating on a variety of platforms can interact, as long as DCOM is available within the environment. To get a secure COM/DCOM architecture component-level testing is important [13]. Three types of properties should be included in a comprehensive component specification from which the functional test cases can be built. Individual *operations* are specified in terms of constraints on their inputs and outputs. These are expressed as pre and post-conditions. The *state* of the object is constrained by an invariant that specifies limits on each of the attributes of the object. Objects protocols which are specified in the state transition diagram define the specific sequence of operations and it should be cheeked correctly, security concerns should be addressed. Object interaction diagram which models interaction between methods and attributes should be well documented so that it should have any security flow.
Methods which implement component operation and component attributes must be handled properly so that interaction happens properly.

5.3 Common Object Request Broker Architecture (CORBA)

Object Management Group (OMG) provides CORBA architecture which allows objects to communicate with other objects and each other irrespective of their location in the network or in the internet [18]. It support multi-tier client server distributed objects approach and data processing require processing additional processing in another computer to require completing the transaction. Objects get a platform which is transparent from location point of view and sharing point of view, because it shares resources. Bugs or errors must be tested before the object or component is integrated into the distributed system, it must be tested in isolation to find and remove the bugs or errors [15]. It is done with unit testing, integration testing [16]. All the ambiguities and inconsistencies must be found out before the implementation of the interface specification [17].

5.4Model-View-Controller (MVC)

Application's data model, user interface, and control logic are separated in three distinct components in MVC; modifications to one component can be made with minimal impact to the others in the MVC. [15].
MVC have three main classes Model, View and Control, which is needed to build an application.

1) *Model*: It is in the domain layer and the domain-specific representation of the information on which the application operates. Domain logic adds meaning to raw data.
2) *View*: Renders the model into a form suitable for interaction, typically a user interface element. MVC is often seen in web applications, where the view is the HTML page and the code which gathers dynamic data for the page.
3) *Controller*: it involves user actions, and invokes changes on the model and perhaps the view.

Data get stored by applications which use a persistent storage mechanism. There is no specific mention for data access layer in MVC because it is understood to be underneath or encapsulated by the Model itself. MVC components are highly interdependent in terms of interface component and this makes very difficult to test MVC. It is very complex to test such component and it requires complex setup to test these components. Separation of concerns is architecture driven and MVC support separation in displaying, updating data and storage and all of them should be tested individually with its own concerns. Frameworks that support user interface and interdependencies are very difficult to test; it requires error prone manual testing and some sort of simulations which simulate user's actions. MVC separates presentation logic and reduction in the number of user interface test cases.

6. SECURITY CONCERNS

Software security concerns are emerging from the first stage of software development even before deciding the particular architecture for the problem domain. All most all the well-known security system Architectures and models, like including Common Object Request Broker Architecture, EJB, COM and DCOM, are considering security as the main issue in the software architecture Since COM/DCOM components have access to a version of the Microsoft Windows API, "bad actors" can potentially damage the user's computing environment. In order to address this problem, Microsoft employs "Authenticode" which uses public key encryption to digitally sign components. Independent certification authorities such as VeriSign issue digital certificates to verify the identity of the source of the component. However, even certified code can contain instructions that accidentally, or even maliciously, compromise the user's environment [42]. Various security threats are present to CORBA like masquerading (attacker pretends to be an authorized user of a system), spoofing and eavesdropping. Besides these integrity violations like trapdoor and viruses can cause problem. Denial of service (because of Flooding), Security of communication between objects, which is often over insecure lower layer communications also pose threat to security.

7. SECURITY RSIK ANALYSIS

We use the term risk analysis to refer to the activity of identifying and ranking risks at some particular stage in the software development lifecycle. Risk analysis is particularly popular when applied to architecture and design-level artifacts [16]. A majority of risk analysis process descriptions emphasize that risk identification, ranking, and mitigation is a continuous process and not simply a single step to be completed at one stage of the development lifecycle. 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) [33, 34].

A prototypical architectural risk analysis approach involves several major activities that often include a number of basic sub steps:-
Learn as much as possible about the target of analysis [35].

- Read and understand the specifications, architecture documents, and other design materials.
- Play with the software (if it exists in executable form).
- Discuss and brainstorm about the target with a group.
- Determine system boundary and data sensitivity/criticality.

- Study the code and other software artifacts (including the use of code analysis tools).
- Identify threats and agree on relevant sources of attack.

Discuss security issues surrounding the software [37, 39].

- Argue about how the product works and determine areas of disagreement or ambiguity.
- Identify possible vulnerabilities, sometimes making use of tools or lists of common vulnerabilities.
- Map out exploits and begin to discuss possible fixes.
- Gain understanding of current and planned security controls.

Determine probability of compromise.

- Map out attack scenarios for exploits of vulnerabilities.
- Balance controls against threat capacity to determine likelihood.
- Perform impact analysis.
- Determine impacts on assets and business goals.
- Consider impacts on the security posture.
- Rank risks.
- Develop a mitigation strategy.
- Recommend countermeasures to mitigate risks.
- Report findings.
- Carefully describe the major and minor risks, with attention to impacts.
- Provide basic information regarding where to spend limited mitigation resources.

During the process of architectural risk analysis, we follow basic steps. Risk management is in some sense fractal. In other words, the entire continuous, ongoing process can be applied at several different levels [40, 41]. The primary level is the project level. Each stage of the validation loop clearly must have some representation during a complete development effort in order for risk management to be effective. Another level is the software lifecycle artifact level. The validation loop will most likely have a representation given requirements, design, architecture, test plans, and so on [48]. The validation loop will have a representation during both requirements analysis and use case analysis. Thus we need economic models of software that take into account costs, benefits, and risks.

8. COST BENEFIT ANALYSIS

Architecture selection and analysis is done with cost benefit analysis and it helps in determining the economic aspects of the software. Cost Benefit Analysis Methods (CBAM) helps in determining the suitability of the software architecture for the problem and solution domain in the software environment [21]. CBAM has many stages and each stage is examined in detail and then implemented so that the cost of the software development can be justified.

The steps of CABM is described in details which helps in evaluating the economic aspect of the architecture, the steps are given below

1. Particular scenarios are chosen and architectural strategies are taken into account for the given aspect.
2. Assess Quality Assurance benefits
3. The architectural strategies' advantages and benefits is quantified.
4. Quantify the architectural strategies' costs and schedule implications
5. Calculate desirability for the architecture
6. Make decisions and evaluate

While choosing the scenario of the concern system security and architectural strategies for the same is considered and are designed that address these scenarios. Architectural strategy should be chosen for the particular scenario like for example, if there were a scenario that called for increased availability, then a particular architectural strategy should be chosen which adds some redundancy and a failover capability to the system [24]. Quality Assurance and quantify the architectural strategies, it elicits benefit information from the relevant stakeholders: Quality Assurance which benefits from business implications and systems conditions (who, presumably, best understand the business implications of changing how the system operates and performs); [47], it helps the architectural strategy. Benefits from the architects (who, presumably, best understand the degree to which a strategy will, in fact, achieve a desired level of a quality attribute) [4]. In the fourth step, Quantify the architectural strategies' costs and schedule implications, it elicits cost and schedule information from the stakeholders. There is no special technique for this elicitation; it assumes that some method of estimating costs and schedule already exists within the organization to achieve it. Based on these elicited values, calculation for desirability for the architecture is carried out. A desirability metric (a ratio of benefit divided by cost) for each architectural strategy is carried out. The inherent uncertainty in each of these values, which aids in the final step, making decisions can also be calculated. The above six steps, helps to calculate the elicited values as a basis for a rational decision making process—one that includes not only the technical measures of an architectural strategy but also business measures that determine whether a particular change to the system will provide a sufficiently high return on investment (ROI) and Suitability.

9. ARCHITECTURE SECURITY MODEL

Security is not easy to build at component level in the software and Architecture Security Model, which is supposed to provide a means for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications, should have to embed the security needs at the architecture level itself [25]. The focus of this methodology seeks to build security concepts into the early stages of the development lifecycle. The model may also be useful for documenting and analyzing the security aspects of fielded systems, and could be used to steer future improvements and modifications to these systems. Fig.2 shows the model used for design of software architecture. This model is elaborated to develop the Architecture Security Model [28]. The various steps of this model are:

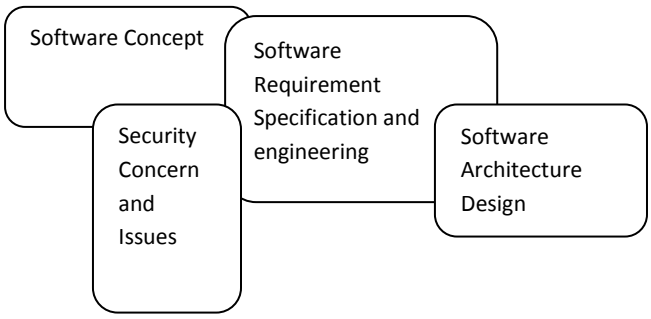

Fig.2 Architecture Design Model

a. **Software:** From the beginning itself it must be clear what is needed and we need to decide what the type of software is under the development process.[30] In the software concept the input passed in this is the statement of the purpose. How it should be done

is by understanding the area of the software in full detail. What is expected output of this module is types of software like system or application, Business software, Engineering/Scientific software, Embedded software, Artificial Intelligence software etc.

b. **Software Security:** This is needed to avoid the unnecessary ambiguous stuff from the security definitions. This step avoids ambiguity in security definitions [31]. The input to this module is candidate definitions and standards and this applies to all types of the software. The methodology adopted for this module is formal structured interviews and forming the focus groups and standards for access of data and sources. After whole process of interviews and focus groups what is expected output is agreed definitions of the standards and security aspects.

c. **Organizational Security Goals:** It's important to identify ideas of security goals of the organization because each organization has its own security requirement and while deciding he architecture for the systems this must be also taken into consideration[32]. Input to this module is: Definitions, candidate goals, business drivers, policies and procedures. Methodology used for deciding the organizational goals is facilitated work session, surveys, interviews in the organization. The expected output is security goals required for the specific organization.

d. **Requirements elicitation Process:** These are required to address problem of scope, of understanding and problem of volatility. While carrying out the requirement elicitation the desired input to this module is definitions, goals, candidate techniques, organizational style, culture, level of security needed, cost benefit analysis etc. How it is carried out is by organizing work sessions. The expected output is Selected elicitation techniques and requirement which are not needed while designing the system.

e. **Developing system artifacts:** This module is required to support elicitation techniques. The expected input to this module is elected elicitation techniques, potential artifacts, threat models, attack patterns. The method used is again by working sessions while finalizing the system artifacts. The desired output is scenarios, misuse cases, models, templates and other important aspect of the system.

f. **Elicit security requirements:** Elicit the requirements using the techniques selected in step 4 and the support of the artifacts developed in step 5. How it is done is by selected techniques and artifacts. Technique used is by carrying out interviews, surveys, model-based analysis, safety analysis, checklists, lists of reusable requirements types, document reviews. The expected output of this module is initial shape of safety and security requirements of the system under the design.

g. **Requirements Categorization:** Categorize the requirements and assess whether they are really requirements or other kinds of constraints. The input given in this module is Initial requirements and architecture. Methods used are by organizing work session using a standard set of categories of the requirement and systems specifications. The expected output is well defined categorized requirements.

h. **Prioritization of the requirements:** Select a method for prioritizing requirements and go through the prioritization process. Deciding the priority of the requirement is also very critical issue in the system design and architectural specification of the particular requirement also plays important role. The input given to this module is categorized requirements. Methods used for prioritization are the methods such as Triage, Win-Win, etc. The expected output of this module is prioritized requirements

i. **Perform risk assessment:** Risk is very vital in the software systems and it is always required to perform a risk assessment activity. There is a wide variety of choices. Input to this module is categorized requirements and targeted operational environment. Methods used for assessing is risk assessment method, analysis of anticipated risk against organizational risk tolerance. The expected output of the system is risk assessment results, added mitigation requirements to bring exposure into acceptable level in the systems.

j. **Requirements inspection:** Finally the requirements must be inspected using a standard inspection or review process to ensure that they are consistent, complete, and testable, and can be achieved or not. Methods used are prioritized requirements, candidate formal inspection techniques, which are used for formal inspection. The methods used for the inspection is inspection method such as Fagan, peer reviews, etc. The expected output is initial selected requirements, documentation of decision-making process and rationale. After all the process adopted in engineering the requirement and risk the rational for security can be established while designing the software.

Each step describes the necessary inputs, the recommended participants and methods to be followed, and the step's final output. The output from each step then flows to the sequential steps that follow. The participants for each step vary depending on the organization under study. Generally, a requirements engineer is tasked with each step, and should consider the input of all relevant stakeholders with respect to the environment of the organization and the study [19,20]. It begins when an organization agrees upon a common base that will serve the methodology to follow. The first task for the organization is to agree upon a common set of security definitions, followed by the definition of organizational security goals. Once the organization has defined a common ground, it can begin to transform its ideas about security goals into actionable security requirements deliverable. Next, the organization chooses from various elicitation techniques, and then can begin documenting important functional information in order to develop artifacts (such as network maps and diagrams, attack tree diagrams, and use and misuse cases). These artifacts are then used to develop initial requirements, which are subsequently categorized to meet the needs of the organization's business goals. Risk assessment allows for the organization to discover how the combination of impact and likelihood of various threats affect the organization's risk tolerance with regard to each categorized requirement. Following this prioritization, a final list of requirements is produced and is inspected by all relevant stakeholders. The final output of this model is a security requirements document

that is designed to satisfy the security goals of the organization [50].
Static analysis approach for the security can also be adopted like ESP, which is a large scale property verification approach, model checkers like SLAM and BLAST which examines program safety and security parameters and some other light weights like Find Bugs can also be used.

10. CONCLUSION

Security issues of the software architectures are discussed in the detail. All the software architectures used for developing the software is discussed in the paper. Security for software architecture is essentially required as it removes loopholes from the system and this can be done by risk analysis. In this approach, business goals determine risks, risks drive methods, methods yield measurement; measurement drives decision support, and decision support drives rework and application quality. We used common concepts in security engineering to create a model for security assessment. The model is based on risk, the most widely accepted form of security measurement. Our approach emphasizes that every part of the system carries a risk, no matter how small. Software components, people and communication channels present security risks. Only a comprehensive analysis of all of them would provide realistic conclusions on the security of a system. Security requirements of the system should be finalized in the architecture design step itself.

11. FUTURE RESEARCH WORK

The architectural specifications must be tested well before the selection of the architecture so the test bed for the same and the test cases must be written for the same. Empirical result and claims can be established for the security claims which are not done in the current paper.

12. ACKNOWLEDGEMENT

The Author is very thankful to the reviewers for giving important feed for the manuscript. Author would also like to thank Sohar University and University of Queensland for their support and encouragement.

13. REFERENCE

[1] Dinesh Kumar Saini, Lingaraj A. Hadimani and Nirmal Gupta “Software Testing Approach for Detection and Correction of Design Defects in Object Oriented Software” Journal of Computing, Volume 3, Issue 4, April 2011, ISSN 2151-9617, Page No. 44-50.

[2] R. Allen, D. Garlan, “A Formal Basis for Architectural Connection” ACM Trans. Software. Engineering. Methodology, 1997. 6(3): pp. 213-249

[3] D. Garlan and M. Shaw, “An introduction to software architecture”, Advances in Software Engineering and Knowledge Engineering, edited by V. Ambriola and G. Tortora, World Scientific Publishing Company, 1993.

[4] D. Gelperin, B.Hetzel,"The growth of software testing" Commun. ACM 31(6), Jun. 1988, pp.687-695

[5] G. McGraw, "Managing Software Security Risks", IEEE Computer, 35(4), March 2002, pp. 99–101.

[6] Dinesh Kumar Saini and Hemraj Saini “Proactive Cyber Defense and Reconfigurable Framework for Cyber Security” International Review on computer and Software (IRCOS) Vol.2. No.2. March 2007, Pages 89-98. ITALY

[7] M. Shaw, R. DeLine, D.V. Klein, T.L. Ross, D.M. Young, G. Zelesnik, "Abstractions for Software Architecture and Tools to Support Them", IEEE Transactions on Sqftware Engineering, 21(4), April 1995, pp.314-335.

[8] K.S.Hoo, J.W.Sudbury, J.R.Jaquith,“Tangible ROI Through Secure Software Engineering”, Secure Business Quarterly: Special Issue on Return on Security Investment, 1(2), A publication of @stake, 2001.

[9] D.Verdon, G.McGraw, "Risk Analysis in Software Design", IEEE Security & Privacy, July/August 2004, pp.32-37.

[10] Dinesh Kumar Saini and Nirmal Gupta “Fault Detection Effectiveness in GUI Components of Java Environment through Smoke Test”, Journal of Information Technology, ISSN 0973-2896 Vol.3, issue3, 7-17 September 2007.

[11] H.Y.Chen, "The design and implementation of a prototype for data flow analysis at the methodlevel of object-oriented testing", Proceedings of the 2002 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2002), IEEE Computer Society Press, Los Alamitos, California, 2002, pages 140–145.

[12] Dinesh Kumar Saini and Nirmal Gupta “Class Level Test Case Generation in Object Oriented Software Testing, International Journal of Information Technology and Web Engineering, (IJITWE) Vol. 3, Issue 2, pp. 19-26 pages, march 2008. USA

[13] Dinesh Kumar Saini and Hemraj Saini "VAIN: A Stochastic Model for Dynamics of Malicious Objects", the ICFAI Journal of Systems Management, Vol.6, No1, pp. 14- 28, February 2008. INDIA

[14] Hemraj Saini and Dinesh Kumar Saini "Malicious Object dynamics in the presence of Anti Malicious Software” European Journal of Scientific Research ISSN 1450-216X Vol.18 No.3 (2007), pp.491-499 © Euro Journals Publishing, Inc. 2007 http://www.eurojournals.com/ejsr.htm EUROPE

[15] P.C.Jorgensen and C.Erickson, "Object-oriented integration testing", Commun. ACM, 37(9), Sep. 1994, pp.30-38.

[16] M.Kolling,J.Rosenberg,"Support for object-oriented testing", Proceedings of the Technology of Object-Oriented Languages and systems, 23-26 Nov. 1998, IEEE Computer Society, Washington, pp.204 - 215.

[17] Y.M. Wang, O.P. Damani, and W.J. Lee,"Reliability and Availability Issues in Distributed Component Object Model (DCOM)", Proceedings of International Workshop on Community Networking", May 1997, IEEE Computer Society, Washington, pp.59-63.

[18] F.J.Hauck, R.Kapitza, H.P.Reiser, A.I.Schmied,"A flexible and extensible object middleware: CORBA and beyond", Proceedings of the 5th international Workshop on Software Engineering and Middleware (Lisbon, Portugal, September 05-06,2005), ACM Press, New York, NY, pp.69-75.

[19] M.Veit, S.Herrmann,"Model-view-controller and object teams: a perfect match of paradigms", Proceedings of the 2nd international Conference on Aspect-Oriented Software Development (Boston, Massachusetts, March 17 - 21, 2003), ACM Press, New York, NY, pp.140-149.

[20] L.Rosenberg, R.Stapko, A. Gallo,"Risk-based object oriented testing", Twenty-Fourth Annual Software Engineering Workshop, NASA, Software Engineering Laboratory, December 1999.

[21] T.M.Khoshgoftaar, E.B.Allen, W.D.Jones, J.P.Hudepohl,"Cost-Benefit Analysis of Software Quality Models", Software Quality Control 9(1) (Jan. 2001),Kluwer Academic Publishers, Manufactured in The Netherlands, pp.9-30.

[22] S.A.Butler,"Security attribute evaluation method: a cost-benefit approach", Proceedings of the 24rd International Conference on Software Engineering, ICSE 2002, IEEE Computer Society, Washington, pp.232-240.

[23] Dinesh Kumar Saini, Jabar H. Yousif, and Wail M. Omar "Enhanced Inquiry Method for Malicious Object Identification" ACM SIGSOFT Volume 34 Number 3 May 2009, ISSN: 0163-5948, USA.

[24] Wail M.Omar, Dinesh K. Saini and Mustafa Hassan "Credibility Of Digital Content in a Healthcare Collaborative Community" Software Tools and Algorithms for Biological Systems in book series "Advances in Experimental Medicine and Biology, AEMB" Springer, Volume 696, Part 8, Page No. 717-724, DOI: 10.1007/978-1-4419-7046-6_73,

[25] Dinesh Kumar Saini "Sense the Future" Campus Volume 1- Issue 11, Page No14-17, February 2011.

[26] Dinesh Kumar Saini and Moinuddin Ahmad "Modeling of Object Oriented Software Testing Cost" The 2011 International Conference on Software Engineering Research and Practice (SERP'11), World Congress in computer Science and Engineering, July 18-21, 2011, Las Vegas, USA. Pp. 333-339.

[27] Dinesh Kumar Saini and Moinuddin Ahmad "Enhanced Software Quality Economics for Defect Detection Techniques Using Failure Prediction" The 2011 International Conference on Software Engineering Research and Practice (SERP'11) World Congress in computer Science and Engineering July 18-21, 2011, Las Vegas, USA, PP. 346-351.

[28] Nr.Mead, V.Viswanathen, Deepa, et.al "Incorporating Security Quality Requirements Engineering (SQUARE) to Standard Life-Cycle Models, Technical notes CMU, 2008.

[29] S L Saini, Dinesh Kumar Saini, and Jabar H. Yousif "Cloud Computing and Enterprise Resource Planning Systems" The 2011 International Conference of Manufacturing Engineering and Engineering Management (ICMEEM-2011), World Congress in Engineering, July 6-9th London UK, PP.681-686.

[30] Dinesh Kumar Saini and Raj Kumar Somani "Malicious objects propagation dynamics in the network", International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), 2011, Digital Object Identifier: 10.1109/ETNCC.2011.5958484, Publication Year: 2011 , Page(s): 47 – 51.

[31] Dinesh Kumar Saini, Lingaraj A Hadimani, Poonam V Vaidya and Sanad Al Maskari "Software Quality Model Six Sigma Initiatives" The 2011 International Conference of Computer Science and Engineering (ICCSE-2011), World Congress in Engineering, July 6-9th London UK, PP. 1226-1231.

[32] Dinesh Kumar Saini, Nebras N. Hasoon, Feras N. Hasoon and Mustafa Hasan. "Review of Query Processing in Distributed Systems", Proceedings of the INFORMATICS 2011 IADIS, International Association for Development of the Information Society, July 20-26, 2011, Rome Italy. Pp 117-123

[33] Dinesh Kumar Saini, Sanad Al Maskari, R G Dabhade, Sandhya V Khandage and Lingaraj A. Hadimani "Broker Architecture for Quality of Service" The 2011 International Conference of Information Security and Internet Engineering, (ICISIE-2011), World Congress in Engineering, July 6-9th London UK, PP 484-490.

[34] Lingaraj A. Hadimani, Dinesh Kumar Saini, Vaishali P Khoche and Sanad Al Maskari, "Comparison of Software and Hardware Design Tools (CASE vs. Simulators)" The 2011 International Conference of Manufacturing Engineering and Engineering Management, (ICMEEM-2011), World Congress in Engineering, July 6-9th London UK

[35] Sanad Al Maskari, Dinesh Kumar Saini, Swati Y Raut and Lingraj A Hadimani, "Security and Vulnerability Issues in University Networks" The 2011 International Conference of Information Security and Internet Engineering (ICISIE-2011) World Congress in Engineering, July 6-9th London UK

[36] Jabar H.Yousif, Dinesh Kumar Saini and Hassan S. Uraibi, "Artificial Intelligence in E-Leaning-Pedagogical and Cognitive Aspects" The 2011 International Conference of Computational Intelligence and Intelligent Systems (ICCIIS-201), World Congress in Engineering, July 6-9th London UK

[37] Nitin B Raut, Jabar H. Yousif, Sanad Al Maskari, and Dinesh Kumar Saini "Cloud for Pollution Control and Global Warming" The 2011 International Conference of Information Engineering (ICIE-2011), World Congress in Engineering, July 6-9th London UK

[38] Dinesh Kumar Saini, N.Hasson, F.Hasson, Mustafa Hassan, "Review of Query Processing in Distributed Systems" Informmatics-2011, IADIS International Conference Italy July 21-26 , 2011.

[39] N.Hasson, F.Hasson, Dinesh Kumar Saini, "Generic Framework for Monitoring Air Pollution in Sohar Industrial Region", ICT, Society and Human Being-2011, IADIS International Conference Italy July 21-26, 2011.

[40] Dinesh Kumar Saini, Sanad Al Maskari and Hemraj Saini, "Malicious Object Trafficking in the Network" IEEE IDCTA-2011, Korea, August 13-16, 201.

[41] Dinesh Kumar Saini, Sanad Al Maskari and Lingraj Hadimani "Mathematical Modeling of Software Reusability" 3rd IEEE International Conference on Machine Learning and Computing (ICMLC,2011)Singapore, February 26-28, 2011, IEEEXplore, 978-1-4244-9253-4/11.

[42] 40. Dinesh Kumar Saini, Wail M. Omar "Software Testing For Semantic Service Oriented Architecture for E-Health Software Services" SERP'10 - 9th international Conference on Software Engineering Research and Practice (USA) http://www.world-academy-of-science.org/, P.No. 240-246

[43] Dinesh Kumar Saini, Osama Abu Rahmeh, H. Saini, Wail M. Omar "Extended Secure Architecture of HIS: HL7" BIOCOMP'10 - 11th International Conference on Bioinformatics and Computational Biology (July 12-15, 2010, USA) http://www.world-academy-of-science.org, P.No. 617-623.

[44] Dinesh Kumar Saini and Hemraj Saini "Achieving Quality Through Testing Polymorphism in Object Oriented Systems,"3rd International Conference on Quality, Reliability and INFOCOM Technology (Trends and Future Directions), 2-4 December, 2006, Indian National Sciences and Academics, New Delhi (India). Conference proceeding.

[45] Dinesh Kumar Saini and Hemraj Saini "Static Code Analysis", NSCOMCS-2005 Proceeding of National Seminar on Mathematics and Computer Science sponsored by UGC.

[46] Dinesh Kumar Saini and Hemraj Saini "Identification and characterization of software testing process for object oriented systems", National Conference on Mathematical Analysis and its Applications in Real - World Problems, Berhampur University, September

[47] Dinesh Kumar Saini and Hemraj Saini "Software Metrics and Mathematical Models in the Software Development Environment for Improving its Quality", National Conference on Mathematical Modeling, BITS Pilani, Oct.2005

[48] Dinesh Kumar Saini and Hemraj Saini "Analytical Study of Mathematical Models For Software Reusability Metrics in Software Development Environment" National Conference on Mathematical Modeling and Analysis – October 2004.

[49] Dinesh Kumar Saini and Hemraj Saini "Statistical Modeling of Extensibility in software" 3rd International Conference on Quality, Reliability and INFOCOM Technology (Trends and Future Directions), 2-4 December, 2006, Indian National Sciences and Academics, New Delhi (India). ISBN 81–7446–434–4 Conference proceeding.

[50] Dinesh Kumar Saini, Lakshmi Sunil Prakash and Wail M Omar "Review of Technological Challenges in Web - Based Learning Content Management Systems (LCMS) with special emphasis on extraction of Learning Contents" International Symposium, College of Applied Science, Ministry of Higher Education, April 13-16, 2010, Oman, P.No. 43-49.