International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 3, Issue 1, July 2014)

A Secure Software Access Measure using Coupling, Complexity and Cohesion Metrics

K.Vaitheki[1], S. Urmela[2]

[1]Assistant Professor, [2]M.Tech Scholar, Department of Computer Science, Pondicherry University, Puducherry, India

***Abstract*— Security being an imperative feature and a crucial need of any software system the security issues have always been secondary for the developers in SDLC progression. The unavailability of the information about the proactive vulnerabilities and the security breaches makes the software much apprehensive. The vulnerability prone nature of the software that affects the secure access that is aimed is identified by adopting the CCC metrics. This paper elaborates Security patterns as reusable solutions and further it paves a way to develop patterns for secure access hence providing explicit solutions to the context of the problem related to secure access. The proposed work has refined the values for the quality attributes related to secure access through careful reviews and map them with necessary code level metrics that affects security at different abstraction levels. The modifications at the code levels based on the internal and external metrics aims to provide a step up in security correlated to secure access.**

***Keywords*-- Security, Patterns, Secure access, SDLC, CCC metrics.**

I. Introduction

Software security is the idea of engineering software so that it continues to function correctly under malicious attack. Software security best practices leverage good software engineering practice and involve thinking about security early in the software lifecycle, knowing and understanding common threats inclusive of language-based flaws and pitfalls, security designing and testing. Software securities need to fit in to the overall concept of operational security and examine some best practices of building security.

Software security best practices and the knowledge of how to tackle need to be explored as there is no clear implication of the design documentations. A security risk may be classified as vulnerability. The time from when the security opening was presented or showed in conveyed programming, to when access was evacuated, a security fix was accessible/ conveyed or the attacker was out of action is the window of vulnerability.

II. The Need For Software Security

The proposed work aims at a lead to develop a general pattern or redefining an existing pattern to achieve the security related issues mainly for a secure access of software. The CCC metrics generically used to access the quality of the software attributes which in turn can be used as a metric related to the identify the factors of vulnerability that affects secure access.

The modifications at the code levels based on the internal and external metrics of the software aims to provide a step up in security correlated to secure access Security patterns available for software that deals with confidentiality, Integrity, authentication, authorization, secure access, non-repudiation, availability and privacy. Security denotes poles apart things with respect to software systems but almost certainly it is associated with four vital key principles.

The metric model that identifies how vulnerable a software becomes the most essential need to avoid further implication at the release stage of the software that may require an enormous reengineering and thereby affect the cost factor vigorously. The extensive researches to identify the various metric models so far do not have a direct impact over the system. Moreover they are much inclined towards measuring the system's dynamic behavior, neglecting the need to evaluate system's quality in the early design phase.

The vulnerabilities that exist do affect the maintainability and testing of the system. Mostly prerelease vulnerabilities which provoke at the testing phase are considered. It will be better to weigh up on the post release vulnerabilities that are proactive in nature and manifest at the operational stage of the software where it also harder to identify the vulnerabilities until they are patented. Pattern users find it hard to locate the required appropriate solution from the steep number of existing patterns.

III. Metrics For Identifying Software Vulnerability

The consequential model provides support for a software quality assessment allied to security. Modifications to support model were done in three steps

1. Identify desirable quality attributes for secure access
2. Select quality-carrying properties from design components
3. Link CCC metrics with the quality attributes related to secure access and provide the mapping of these attributes with the metrics.

The design properties are derivative of the design components such as services, operations, messages used in operation calls, and connections among services.

When identifying and designing services, these design components have to be appropriately combined to qualify a good design.

The Coupling, complexity and Cohesive measure the quality of the software attributes and more than it branches out to locate the vulnerabilities of the system at the operational level of the software proactively.

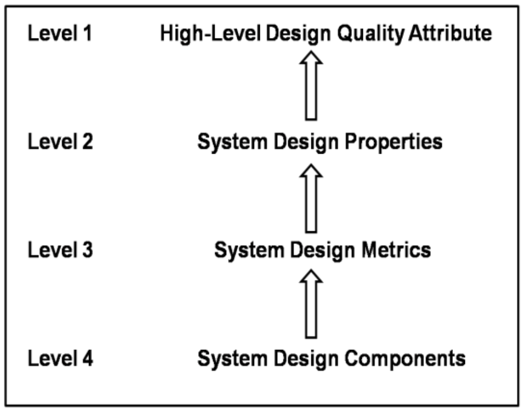


Fig 1 Operational level of the software

IV. ARCHITECTURE DESIGN

Service oriented architecture is a very popular architecture paradigm for designing and developing distributed system. It is as information technology approach in which applications make use of network services that are available over the internet.

It is an architectural style where system consists of service consumers and service providers. An architecture style defines a vocabulary of components and connectors type and constraints on how they can be connected.

Table 1.
System's Components And Connectors

Basic component	Service users
	Service provider
Auxiliary component	Enterprise service bus
	Directory service
SOA connector	Synchronous calls
	Asynchronous calls

The constraints that apply to the SOA architectural style are

A. Service Provider

The service supplier is the system addressable element that acknowledges and executes demands from consumers. It could be a mainframe system, a part or some other sort of programming framework that executes the administration demand .The administration supplier or the service provider distributes its agreement in the registry for access by administration shoppers.

B. Service Consumer

The service consumer is a requisition administration, or some other sort of programming module that obliges an direction. It is the element that launches the spotting of the service in the registry tying to the administration over a transport, and executing the service capability.

C. Service Registry

A system based catalog that holds accessible administrations is nonentity but the service registry. It is an element that acknowledges and stores contracts from administration suppliers and gives those agreements to the intrigued service consumers.

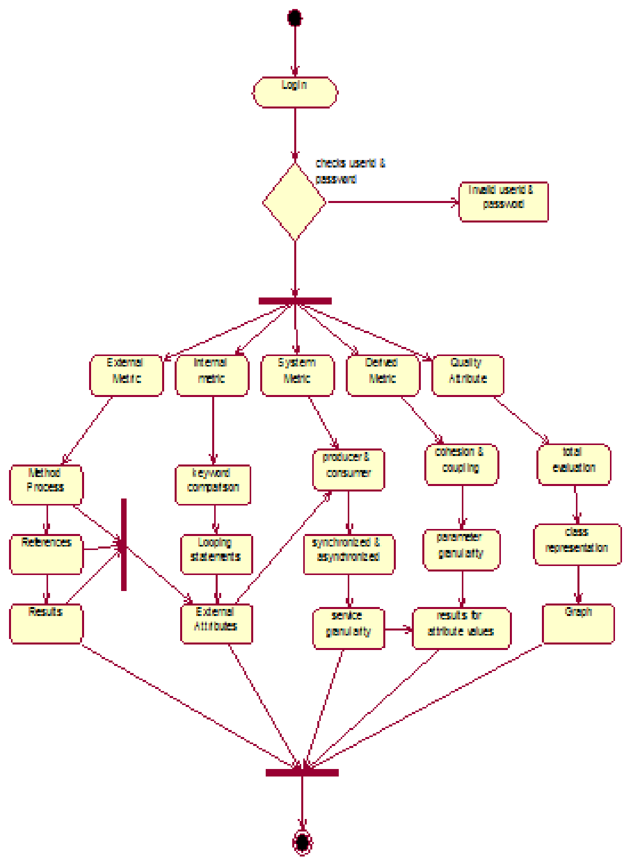


Fig 2. Activity Diagram of the entire system

D. Service Contract

Service contract is a particular of the way a consumer of an administration will collaborate with the supplier of the administration. It defines the arrangement of the appeal and reaction from the administration. It may require a set of preconditions and post conditions that indicate the state that the service must be into execute a specific capacity. The agreement might additionally point out nature of service (QOS) levels. QOS levels are determinations for the non practical parts of the administration. Case in point, a nature of administration characteristic is the measure of time it takes to execute a service technique.

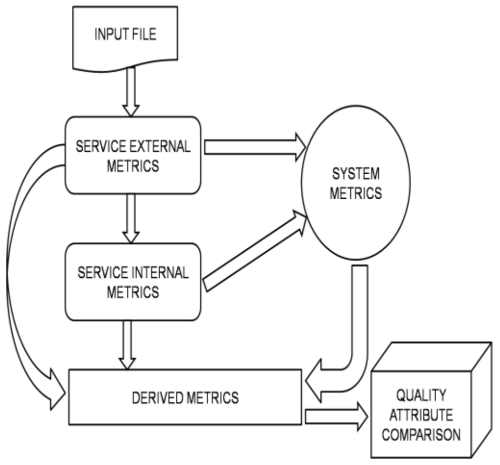


Fig 3. The Quality Attributes Metric Paradigm

V. IDENTIFY SOFTWARE INTERNAL AND EXTERNAL METRICS

A. Service Internal Metrics

Identify code software and analyze it thoroughly with a testing tool which is used to hit upon the vulnerability of Software at the code level. The service internal metrics use service internal elements such as service name, operations provided by the service, and characteristics of the messages defined in the service. This module computes number of operations, numbers of Fine-grained Parameter Operations, number of Messages used, number of asynchronous operations, number of synchronous operations, number of inadequately named operations in the given services. Number of operations is used to measure complexity and is adapted from number of methods metric in systems. Cohesion is a property which implies the degree of relationships between operations defined in a service and is measured by the metric defined as average used message factor. This metric is inversely proportional to the average number of messages in a service because smaller number of messages.

Since the fine-grained service is defined that the service which perform single function, it is assumed that fine-grained parameter operations are operations which are having single parameter. “Number of message “is calculated by addition of total number of producer services and total number of consumer services.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 3, Issue 1, July 2014)

Web Service operations might be either synchronous appeal reaction or offbeat restricted synchronous solicitation reaction (the default conduct) implies that each time a customer provision summons a Web Service operation, it gets a SOAP reaction Asynchronous restricted implies that the customer never gets a SOAP reaction, even an issue or exception when outlining non-concurrent one-way Web service operations.

The backend segment that actualizes the operation should expressly return void. The out or in-out parameters to the operation cannot be specified whereas the in parameters are specified .So the operation which have return type are assumed as synchronous operations and the operation which have return type as void are assumed as asynchronous operations. All the constructors and destructors are assumed as Inadequately Named Operations.

B. External Metrics

The External metrics use information from services it is connected to. Metrics in this group are used to measure the characteristics of consumer and producer services either directly or indirectly connected to a given service.

The external metrics is the computation of number of Consumers in same level, number of directly connected producer services, and number of directly connected consumer services, total number of provider services and the total number of consumer services in the system.

Table 2.
System Metrics And Rationalization

Metrics	Rationalization
Coupling	The strength of dependency between services in system
Cohesion	The strength of relationship between operations in a service
Complexity	Measures the difficulty of understanding relationship between services
Design size	The size of the system design
Service Granularity	The appropriateness of size of services
Parameters Granularity	The appropriateness of size of parameters
Consumability	The likelihood of other services to discover the given service

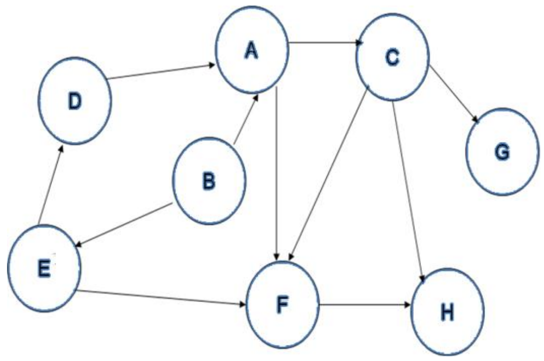


Fig 4. Service Composition Diagram

The Service Composition shows connection between different numbers of services in a file.

Number of Directed connected Consumer Services NDCS (A):2 (F&C)

Number of Directed connected Producer Services NDPS(A):2 (B&D)

Total number of Producer Services NTPS(A): 5 (B,D , E,F,C)

Total number of Consumer Services NTCS(A):4 (F,C,H,G)

C. Relationship between Design properties and Derived Metrics

Table 3
System Metrics And Clarification

System metrics	Clarification
SSNS	System Size in Number of Services
NINS	Number of Inadequately Named Services
NINO	Number of Inadequately Named Operations
TMU	Total Number of Message Used
NAO	Number of Asynchronous Operations
NSO	Number of Synchronous Operations
NFPO	Number of Fine-Grained Parameter Operations
NPS	Number of Process Services
NIS	Number of Intermediary Services
NBS	Number of Basic Services

VI. Implementation And Results

The proposed work has utilized metrics and the attributes related to evaluate design quality in the design phase, giving organizations a chance to find and repair problems before they find their way into the working system excluding a huge aggregate of latent disbursement for problem resolution. The Complexity, Coupling and Cohesion metrics with respect software code behavior is considered and result is exposed through a graphical Structure related to Secure Access Attributes.

A. Identifying an Software

Identify software and evaluate it thoroughly with an testing tool that identifies the vulnerability of a software. A security danger may be considered as weakness. The utilization of vulnerability with the same significance of danger can prompt perplexity. The danger is fixed to the potential of a misfortune. At that point there are vulnerabilities without danger, for instance when the influenced stake has no quality. The vulnerability with one or more occasions of working and completely actualized assaults is considered an exploitable vulnerability- a weakness for which an exploit exists. The window of weakness is the time from when the security gap was presented or showed in sent programming, to when access was evacuated, a security fix was available/deployed, or the attacker was cripple.

B. Analyzing an Software Internal and External Metrics

The service internal metrics use service internal elements such as service name, operations provided by the service, and characteristics of the messages defined in the service This module computes number of operations, numbers of fine-grained parameter operations, number of Messages used, number of Asynchronous operations, number of synchronous operations, number of Inadequately named operations in the given services.

The External metrics use information from services it is connected to. Metrics in this group are used to measure the characteristics of consumer and producer services either directly or indirectly connected to a given service.

C. Analyzing Overall System Metrics

The last group, system metrics, measures the characteristics of the entire system in general such as Usability, Reliability, Reusability, Maintainability, Portability. Metrics attributes and quality characteristics have the following relations: Functionality: this is the capability of software can meet user's requirement and run stably in expect environment.

Here the capability means clear or unclear functions of software. So this characteristic is used for description of what the software will do in order to meet user's requirements.

D. Quality Attribute Factor

Anti-patterns may be technical, or more related to general software processes and projects. When detecting them in order to uncover potential quality problems in software architecture, just the technical class of anti-patterns is applicable. Finding an instance of an anti-pattern from an architecture typically means that the architecture should be improved by refactoring the suspicious instance into a more robust form – e.g, into an instance of a design pattern. So the design of a graphical model of an system will deals that how an software can matches related to the attributes of secure access.

1) *Effectiveness* The degree of business requirements reflected in the design.
2) *Understandability* A measure of the effort necessary to learn or comprehend the design.
3) *Flexibility* The ease of changing the previous design to accommodate new functionalities.
4) *Reusability* Measures how much of the design allows reapplication to other solutions.
5) *Discoverability* The likelihood of other services to discover the given service.

E. Relationship between Derived Properties and Derived Metrics

i. Coupling

Coupling was originally defined as the measure of the strength of association established by a connection from one module to another. Most of the existing techniques and measuring coupling metrics are classified by procedural programming and object-oriented programming.

Table 4
System Metrics Varaibles (Coupling) And Explanation

System Metrics (Variables)	Rationalization	Values
Dbase	Total number of Base Class Used in the Program	1
Dcons	Total number of Derived Class used in the program	0
Service	No of time the Class Called	1
Coupling= (base class constructor + derived class constructor)/ service = 1		

ii. Cohesion

Cohesion was originally defined as a measure of thedegree to which the elements of a module belong together. In a highly cohesive module, all elements are related to the performance of a single function. They are categorized into Coincidental, Logical, Temporal, Procedural, Communicational, Sequential, and Functional.

Table 5.
System Metrics Varaibles (Cohesion) And Explanation

System Metrics (Variables)	Rationalization	Values
tmu	Total number of messages used	1
Service	No of time the Class Called	1
iaum	(base class constructor + derived class constructor)/ service	1
Cohesion = service / Total number of messages used =1		

iii. Complexity

The complexity metrics identified a relationship between complexity of a service and amount of time required to build such a service.

Measure the difficulty of understanding relationship between services, but looks at the individual complexities of each of the composed Services Metrics are first computed for individual services and then compiled into a more global metric.

The link between complexity and security is a well-accepted fact in system security engineering. In particular two of these design principles, namely the principle of “psychological acceptability” and the principle of “economy of mechanisms”, directly relate to the issue of complexity. The principle of “psychological acceptability” states that the introduction of a security mechanism should not make the system more complex than it is without it.

Table 6.
System Metrics Varaibles (Complexity) And Explanation

System Metrics (Variables)	Rationalization	Values
no	Complexity	3
tso	Asynchronous operation	1
tao	Synchronous operation	1
complexity = (asynchronous operation + synchronous operation) * 1.5 =3		

iv. Design

The objective of user interface configuration is to make the user's association as regarding achieving client objectives what is frequently known as user-centered design. The immense user interface outline encourages completing the assignment nearby without attracting unnecessary consideration regarding it. Visual computerization may be worn to help its convenience.

Table 7
System Metrics Varaibles (Design) And Explanation

System Metrics (Variables)	Rationalization	Values
Ns	Design Size	2
Service	No of time the Class Called	1
design size(ns)= service + 1= 2		

v. Service granularity and Parameter granularity

It measures the appropriateness of size of services. Web administrations based SOA have a tendency to be coarse-grained benefits that perform a related set of business capacities rather than a single task are said to be coarse grained Services that perform a single operation are said to be fine grained.

The idea of granularity applies to services in two ways. First and foremost, it is connected to the extent of space the whole service executes. Second, it is connected to the extent of the area that every strategy inside the interface actualizes.

Table 8
System Metrics Varaibles (Service Granularity) And Explanation

System Metrics (Variables)	Rationalization	Values
Tso	asynchronous operation	1
Tao	synchronous operation	1
x /= service = /1=2		
Service	No of time the Class Called	1
y /= service = 2/1		
aomr(Service Granuality) = x / y= 4/4=1		

Table 9.
System Metrics Varaibles (Parameter Granularity) And Explanation

System Metrics (Variables)	Rationalization	Values
tso	asynchronous operation	1
tao	synchronous operation	1
cpr /= (tso + tao) =1/2=0.5		

vi. Consumability

x = service - iaservice=1-0=1
y = tso + tao–iaoperation= 1+0=2
y = y / ((tso + tao) * 2) = 2/(1+1)*2)=2/(2*2)=2/4=1/2=0.5
ansor = x + y =0.5+0.5=1=Consumability
Consumability=1

VII. Secure Access Quality Attribute Factor Computation

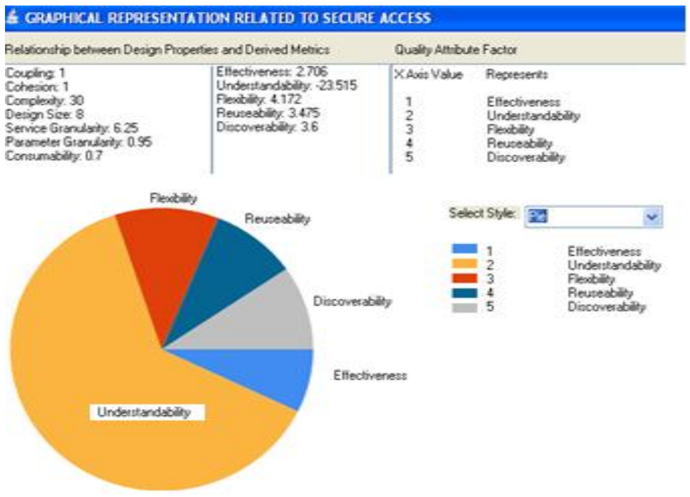
Effectiveness = 0.33 * Cohesion + 0.33 * Service Granularity + 0.33 * Parameter Granularity
= 0.33*1+0.33*1+0.33*0.5 =0.825

Understandability = -0.66 * Coupling + 0.25 * Cohesion - 0.66 * Complexity -0.66 * Design Size +0.25 * Service Granularity + 0.25 * Parameter Granularity + 0.25 * Consumability
=-0.66*1+0.25*1-0.66*3
0.66*2+0.25*1+0.25*0.5+0.25*1
=-3.085

Flexibility = -0.22 * Coupling + 0.61 * Service Granularity + 0.61 * Parameter Granularity
= -0.22*1+0.61*1+0.61*0.5 =0.695

Reusability = -0.5 * Coupling + 0.5 * Cohesion + 0.5 * Service Granularity + 0.5 * Consumability
= -0.5*1+0.5*1+0.5*1+0.5*1 = 1

Discoverability = 0.5 * Service Granularity + 0.5 * Parameter Granularity
= 0.5*1+0.5*0.5 = 0.75


GRAPHICAL REPRESENTATION RELATED TO SECURE ACCESS

Relationship between Design Properties and Derived Metrics

Coupling: 1
Cohesion: 1
Complexity: 30
Design Size: 8
Service Granularity: 6.25
Parameter Granularity: 0.95
Consumability: 0.7

Effectiveness: 2.706
Understandability: -23.515
Flexibility: 4.172
Reuseability: 3.475
Discoverability: 3.6

Quality Attribute Factor

X Axis Value	Represents
1	Effectiveness
2	Understandability
3	Flexibility
4	Reuseability
5	Discoverability

Fig 5. Graphical Representation of Secure Access Quality Attribute

VIII. Conclusion And Future Enhancements

As per today's requisite security is to be initiated at the initial phase of software development to put off major problems that may crop up at the implementation phase as so this paper covers various security issues in software development stages .It describes that how the quality attributes related to secure access can be useful in the help of security patterns for developing secure systems. The Overall goals of our project are to allow the users to access and manage the quality of service explicitly from the early stages of system development. The new system addresses the problems to prevent the waste of effort which would otherwise be magnified in later stages. The use of CCC metrics enables us to quantify system quality and realize how menacing it would be to make use of the metric that gets inferior at the stage of newer vulnerability releases that compromises security. It is an attempt to improve the secure access of a system and reduce a static gap that prevails between the availability, production and utilization of security patterns. In future more and more security patterns need to be developed and developers must understand the need of security parameters and embrace software security best practices throughout software development process correlated to secure Access.

The Complexity, Coupling and Cohesion metrics with respect software code behavior is considered and result is exposed through a graphical Structure related to Secure Access Attributes. The proposed work has utilized metrics and the attributes related to evaluate design quality in the design phase, giving organizations a chance to find and repair problems before they find their way into the working system excluding a huge aggregate of latent disbursement for problem resolution. The system can be enhanced to measure the other quality attributes so as to evaluate the design of any software application wherein any files can be evaluated through adding some additional components and tools, program execution can be shown graphically by using software tools, forums, Local Language support and patterns related to Secure Access for software.The idea of the work can be further extended to propose a framework for identifying the appropriate patterns for a software security related to Secure Access. It is feasible through analyses of the bug report by adopting statistical and data mining techniques where the CCC metrics can act as vulnerability indicators.

REFERENCES

[1] Bigger staff, Ted. “Design Recovery for Maintenance and Reuse.” IEEE Computer 22, 7 (July 1989): 36-49.[Brooks 75] Brooks, F. The Mythical Man-Month—Essays on Software Engineering.

[2] R. J. A. Buhr, R. S. Casselman,” Use case maps for object-oriented systems”, ISBN: 9780134565422, Prentice Hall India

[3] M. Schumacher et al., Security Patterns: Integrating Security and Systems Engineering, John Wiley & Sons, 2005.

[4] Debra.S.Herrmann , “ Complete Guide to Security and privacy metrics” , Auerbach publications, taylor&francis group.

[5] Nabil Fakhfakh, Herv´eVerjus, Fr´ed´ericPourraz,,PatriceMoreaux “Measuring The Satisfaction Degree Of Quality Attributes Requirements For Services Orchestrations”,CTRQ 2011 : The Fourth International Conference on Communication Theory, Reliability, and Quality of Service, ISBN: 978-1-61208-126-7 pp - 89-94.

[6] Maxim Schnjakin, Michael Menzel, ChristophMeinel,” A Pattern-driven Security Advisor for Service-oriented Architectures” , *SWS’09,* November 13, 2009, Chicago, Illinois, USA.pp: 13-20.

[7] S. S. Yau, N. Ye, H. Sarjoughian and D. Huang, *Arizona State University, Tempe, AZ 85287-8809, USA*” Developing Service-based Software Systems with QoS Monitoring and Adaptation” .

[8] Heather Hinto, Maryann Hondo ,Dr.Beth Hutchinson ,” Security Patterns within a Service-Oriented Architecture”, November 2005, http://www.ibm.com/websphere/developer/services/.

[9] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua ,Philippe Comte, PålKrogdahl, Min Luo Tony Newling,” Patterns: Service Oriented Architectureand Web Services”, ibm.com/redbooks.

[10] Liam O’Brien , Len Bass Paulo Merson, “Quality Attributes and Service-Oriented Architectures”, Technical Note CMU/SEI-2005-TN-014.

[11] Darrell M. Kienzle, , Matthew C. Elder, . David Tyre, James Edwards-Hewitt, “Security Patterns Repository Version 1.0”.http://www.securitypatterns.com.

[12] Liam O’Brien, Len Bass, Paulo F. Merson, “Quality Attributes and Service-Oriented Architectures”, Carnegie Mellon University, Research Showcase ,Software Engineering Institute, 9-1-2005.

[13] Yanguo (Michael) Liu, IssaTraore,” Complexity Measures for Secure Service-Oriented Software Architectures”, www.cs.unh.edu/~it666/reading.../complexity_leads_insecurity.pdf.

[14] Fernando Brito e Abreu ,RogérioCarapuça, “Object-Oriented Software Engineering:Measuring and Controlling the Development Process”, Revised version: Originally published in Proceedings o "4th Int. Conf. on Software Quality", October 1994.

[15] Hamid Mcheick, Yan Qi ,” Quality Attributes and Design Decisions in Service-Oriented Computing”, 2012 International Conference on Innovations in Information Technology (IIT), pp 283-287.

[16] Kai Qian, Jigang Liu, Frank Tsui ,” Decoupling Metrics for Services Composition” 5th IEEE/ACIS International Conference on Computer and Information Science, 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. Pp 44-47.

[17] Pham ThiQuynh, Huynh QuyetThang,” Dynamic Coupling Metrics for Service – Oriented Software”, International Journal of Computer Science and Engineering 3:1 2009.

[18] Philip Bianco, Rick Kotermanski, Paulo F. Merson, “Evaluating a Service-Oriented Architecture”, Carnegie Mellon University, Research Showcase, 9-1-2007.

[19] MamounHirzalla, Jane Cleland-Huang, and Ali Arsanjani, “A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures”, ICSOC 2008, LNCS 5472, pp. 41–52, 2009. © Springer-Verlag Berlin Heidelberg 2009.

[20] Mikhail Perepletchikov*, Caspar Ryan, Keith Frampton, and Heinz Schmidt,” Formalising Service-Oriented Design”, JOURNAL OF SOFTWARE, VOL. 3, NO. 2, FEBRUARY 2008.

[21] Claudia Steghuis, “ Service granularity in SOA Projects : A Trde off Analysis”, University of twente.

[22] Si Won Choi Soo Dong Kim “A Quality model for evaluating reusability
of services in SOA”, IEEE,2008

[23] http://docs.oracle.com/

[24] http://academic.research.microsoft.com/Paper/5132835.aspx