Cloud Computing Security for Organizations using Live Signature – TPALM Printing Client Service

Atif Farid Mohammad, Emanuel S. Grant
Department of Computer Science, University of North Dakota, Grand Forks, ND USA
atif.mohammad@und.edu, grante@cs.und.edu

Abstract

Cloud is taking over the computing environment in both public as well as private sector. This has increased the use of service-oriented architecture (SOA) for the development of services later deployed in the Cloud. This paper presents a Cloud Security algorithm using SOA 3.0 for secured transactions on the data, which usually governments of countries like USA International Traffic in Arms Regulations (ITAR) and Export Administration Regulations (EAR) requires to be utilized and distributed only within United States by security cleared personal only. In this paper, we describe a novel algorithm and corresponding cloud service as Cloud Monitoring Gateway (CMG). The current service prototype simulates the behavior of actual Cloud Security Gateway Application (CSGA) using the algorithm called as TPALM (The Privacy Authentication Latency Management). This simulation is coarse-grained, but is capable of measuring the privacy authentication on the given variables of a legit user. We also present an evaluation of this service utilization on actual data.

Key Words: Service, SOA, Software Engineering, System, Cloud Computing, QOS, Data Centre,

1. Introduction

Cloud refers to the stipulation of different services required by users on-demand in the form of computational resources as a combination of services. The services are designed and developed using service oriented architecture or SOA. Any software application that requires frequent modification needs to be separated from the servicing applications, which is consistent and rarely needs to be updated. Service Oriented Architecture (SOA) is the application of this understanding on the knowledge management of a business.

Let us take an example to explore SOA. Each puzzle piece given in Figure 1.1 is a service provided by a retailer, travel agency, bank or a government service provider. These services are available around the world, as an offshoot of global business. Figure 1.2 depicts the servers providing these distributed services across the globe. Using the Cloud, these services can be accessed almost anywhere in the world. The magic of SOA works for consumers as well as industrial internal-operational and managerial users. The use of SOA generates the structure of these services shown in Figure 1.3.

Figure 1.1: Services

Figure 1.2: A distributed view of services [1]

SOA evolved in stages over the last few decades, since industrial automation increased. The services we use today process requests as input and produce output for customers, other systems or services. These systems or services orchestrate the data when generating messages among each other and to us as the user. Each service is owned and governed by a business entity and works within a certain body of rules, defined by the policymakers.

SOA characterizes and provides the composite framework for the Cloud services based on open standards. Multiple services [2] can be combined to serve a consumer at one independent Cloud; for example, a travel agent can provide car rental, hotel reservations or can book an entire vacation. This is a convenient way for the consumer to get several services with one phone call or one website utilization. Expedia [6] can be taken here as an example. The services designed with transparency to the background-used technology and business processing management rules, for all internal and external users, offer the highest ongoing returns in terms of use and feedback for the industry's investment of skilled personnel's time.

2. Background

The challenges of the present financial climate can be resolved by software engineers and decision makers, such as CTOs or MIS managers, by aligning business needs, using service components to improve service to consumers. SOA is a boon to an enterprise looking to create service components in an agile fashion and reuse an existing infrastructure. According to Schreiner and Lamb [3], systems of the future will be based on the concepts of SOA. Service applications will be composed of a number of

DOI: 10.5176/2010-3043_2.3.179

© 2012 GSTF

individual services running in the Cloud. As illustrated by Erl [4], service component application logic can be divided into two levels: a service interface, where loosely coupled services are available with their implementation and technology platform; and a service-using application level in which service application logic is developed and deployed on different technology platforms. These services communicate via open protocols.

With SOA, these services can be combined software as a service or SaaS at one front-end platform on a website to provide the "property for sale" information or other services as mentioned above. To make this clear, we can take the example of Amazon's [5] store-front. Customers use a browser to get the displays on Amazon.com. The front-end website infers the customer's intent and triggers the services that do things like acquiring the data for the current on-sale products, or getting the customer's order. The important thing to note here is that the servicing components do not make proposals to the customers, and these services have no idea who they are talking to. These services are serving customers by getting data from some other services, which might be residing at some other server and can provide only product details. Some other services might be obtaining customer order details to invoke other services for shipping the products.

3. Related Work on Cloud Security Issues

As per Schneier, et. al, "Security is not a product - it's a process." [7] Service security in the Cloud is not only a quality attribute, it is also regulated by governmental laws. There are several laws that can be identified to understand the need for Cloud security from the perspective of service providers, and from the perspective of the users. Some of the computer related crimes that are addressed by Criminal Laws [8] are:

- Unauthorized access, Exceed authorized access, Intellectual property theft or misuse of information, Child pornography
- Theft of services. Forgery, Property theft (i.e., computer hardware, chips, etc.), Invasion of privacy,

Cloud computing needs security of the cloud tested at an extensive level. To test any of the cloud service we need to know the base architecture of the Cloud security service(s). Service-Oriented Architectures (SOA) presents an advanced architectural concept with significance. Dorner et al. [9] have brought forward a few considerations of SOA in terms of End User Development (EUD). The authors' analysis is based on requirements for EUD systems and empirical studies, taken from earlier research work [10]. Dorner et al. have suggested in their study, that SOAs can be extended with structures for in-use modifications; the design of user-adaptable next-generation systems is also possible. EUD can also be suited develop Cloud service for the purpose of securing end-user as well.

With SOA-provided flexibility, the new tailorable systems can be produced, and platform independence can also be achieved. Services designed using SOA are formulated software applications, and this formulation is closer to business domains. Cloud computing embeds almost all known computing devices as well as several of software, such as SaaS (Software as a Service) [1], PaaS (Platform as a Service) and several Operating Systems. It also utilizes as many data communication networks, such as local area networks (LANs), metropolitan area networks (MANs) and wide area networks (WANs). A recent survey by Cloud Security Alliance (CSA) & IEEE indicates the eagerness of corporate sectors to adopt cloud computing, major bottle neck is the security is needed both to hasten cloud adoption, while making sure to achieve regulatory drivers coverage for their daily activities. It is vital for organizations to look critically at security models to examine the confidentiality issues for their business critical tactless applications. Due to several such gaps, it is yet not possible to provide guarantees that corporate data in the "cloud" is secured, if not impossible, as they provide different services like SaaS, PaaS, and IaaS. Each service has its own security issues [11].

In SaaS, the client has to depend on the provider for proper security measures [12]. Next section details relationship of Service Oriented Architecture SOA 3.0 methodology relationship with software engineering. This discussion provides a base for an understanding of the Cloud Security Model 2.0 novel design.

4. SOA 3.0 and Software Engineering

Although software engineering has progressed in the past few decades, there are two realities still dominating the lives of real-world software engineers [13], whether it is traditional software applications development or a SOA service development, these two realities are:

A. The maintenance phase still costs typically 40% to 80% to an organization.

B. The existing legacy systems need to be maintained, and this maintenance involves the need to understand the existing legacy system.

Software engineers perform several types of tests to extract data; which is then processed. Suitable or non-suitable results are recorded during and/or after the experiments. This type of data processing, called conceptual modeling [14], [15] is performed by software QA testers. "*In a few systems, the conceptual view plays a primary role. The module, execution and code views are defined indirectly via rules or conventions, and the conceptual view is used to specify the software architecture of a system, perhaps with some attributes to guide the mapping of other views* [16]."

© 2012 GSTF

5. Proposed Cloud Security Model

It is vital that the use of cloud is to be secured to prevent any security breaches for both users as well as service providers [17]. In this research paper, we present a Cloud Security Architecture using Cloud Security Model (CSM 2.0) to make sure that both data and user requiring this data is protected under the federal law of ITAR/EAR [18][19]. This model is based on Layered Architecture and looks into a Cloud domain through the Resource Management as shown in the given figure.

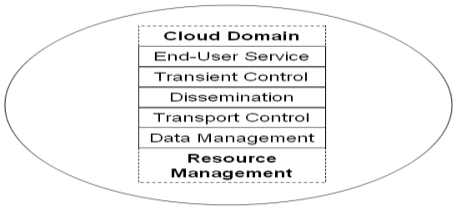


Figure 5.1: Cloud Security Architecture [17]

The proposed architecture can be understood by the following Figure 5.3. CSM 1.0 contains a consumer login service, which needs to adhere the procedures and protocols provided by the service/service provider. Every service has an end-point; we call it Edge, which is to bind the consumer to use the service on the basis of contract accepted by both consumer and service provider. Communication among user or consumer is the key transient control between both for transparent use of the service(s). These procedures and processes are depicted in Figure 5.2.

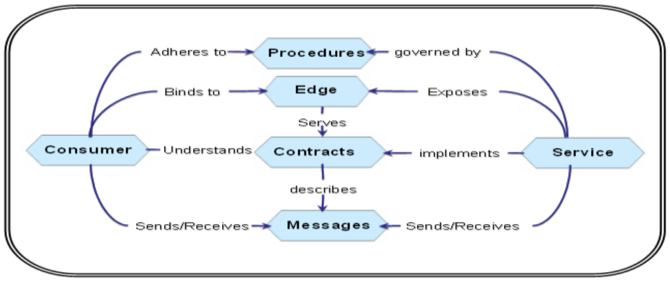


Figure 5.2: Cloud Security Model – CSM 1.0 [17]

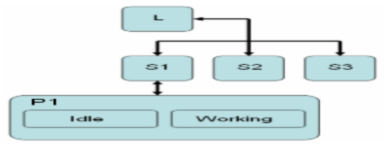


Figure 5.3: A functional example of an SRD

The processing operator **P1** connects and relates elements to show the logical flow to construct a block in order to illustrate the performance of a desired system. The performance of a system is illustrated in two foremost ways, expressed by the combination of communicational links and conditional operators. Let us assume the notations as shown as L “List of Services”, S1 ”Service 1”, P1 “Process Operator 1”, P1.1 “Idle Service”, P1.2 “Working”.

L is a **list of services** available in the Cloud containing a combination of **S1**, **S2** and **S3** (the **services** provided by a service provider) and **D1** (a service provided by a data centre in the Cloud). There can be two major processes a service can be in—0 and 1. The service is **Idle** or **Working**. The use of CSM 1.0 protocols provides us a secured way to use any further services from this point onwards.

Let us consider the service S1 is a Security Check Point 1 and the user’s authentication is clarified for further processing. The service might be idle due to no job being needed to be done at this moment in time.

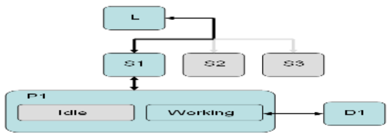


Figure 5.4: A working service prototype

The working service might also be idle, as there can be a delay in receiving some messages for next level of authentication combining user’s as well as service provider’s authentication for other services or a data centre of some kind. P1.1 and P1.2 can further be drilled down. The Figure 5.4 shows a prototype of getting required information for the service user of **L** from data centre service **D**.

The service **S1** as shown in Figure 5.5 is the actual service built as an application block. This service block S1 connects with another service, **D1,** which is a service provided by a data centre in the Cloud and is assigned with first encrypted key **K1**, upon third level of authentication key **K3,** which is to be assigned by Dissemination level as the main locked layer key **K2** to issue clearance to get the required data for the user of service **L**. The processing block as shown in earlier Figure 4.4 can be seen as an independent processing operator connecting two services by communication of the requestor’s requirement, and the operator gets the resultant data set(s) and delivers it to the requestor.

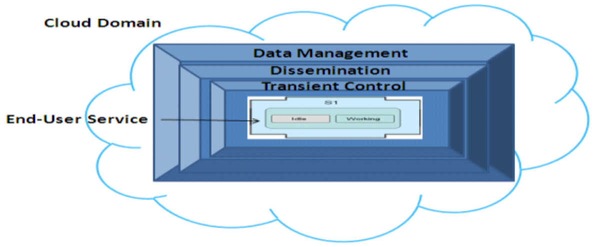


Figure 5.5: Three layered Cloud Security Model 2.0

Let us look into further details of these services; **L** is a listing service of several services. For simplicity’s sake, we will take the service **L** as given below:

- Detailed display of services available by a provider has applied CSM 1.0 protocols.
- Facilitates user’s request as input to get a resultant data set at each entry and exit point of the service.

© 2012 GSTF

The service **S1** provides the following features:

- Generates a query on the request input once cleared from CSM 1.0 protocols, it stores the data for future feedback for service designer and manages bandwidth rates of data set receiving, by delivering the data set(s) to service **L** upon clearing from CSM 1.0 protocols.

The service **D1** provides the following features:

- Receives query from **S1** cleared by CSM 1.0 protocols, also processes the resultant data set(s), it delivers to **S1** upon clearance from CSM 1.0 protocols and stores the query for future use for some other user.
- Stores query snapshot for feedback for service designer & Self-manages the storage usage, by managing bandwidth rates of every query.
- K1 contains detail of data center originating IP address, as well as physical address with service provider's name and company license info.
- K3 contains detail of requestor/user's originating IP address.
- The main process of K2 is to confirm that both user and data provider are with ITAR/EAR defined boundaries;

Next session presents a novel algorithm with linear analysis conducted to prove that it is viable new solution, which can be adapted by Client side services to get secured sessions to work with.

6. Client Service Algorithm & Linear Analysis

A Service user ID is the SPID, IP, and MAC, which are computer ids used for networking. It is important to know that some of the variables, such as ACCEPTED in the following write-up, are derived from a previous function or process mentioned in the algorithm. All incoming and outgoing messages are encrypted and decrypted respectively. Notations are shown by a number followed by a character '**s**' means Cloud Server and '**c**' means Client Service. Encrypt_#char(DATA) & Decrypt_#char(DATA) represent different encryption and decryption methods used in this program. The number mentioned here is a different encryption and decryption. The character is the source of the encryption.

The Time Analysis is done using variables Cost (c#), Times (t#) (number of times executed), With pseudo code proof. Travel time from Client Service to Cloud Server is represented by '**t**'. Times are marked with letters other than are ranged representations depending on programmer's preference will be noted in the table given. Capital characters represent algorithms stated below. Encryption and decryption algorithms run between **ϴ(n)** and **O(n!)** and are marked as **e(c or s)#** and **d(c or s)#** the '**c**' and '**s**' represent client and server respectively.

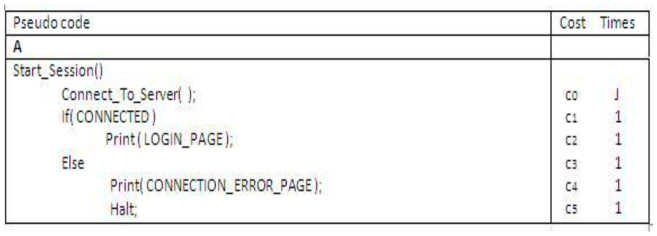
6.1. Mathematical proof: We have used discrete mathematics and a summation notation to describe the process steps in mathematical form. $\sum_{Start}^{End} Process$ describes a process where there is most likely a loop and starts at some point increments in some way bit block etc. and ends at the end of the data. In short it is a sum of the steps used for each data size. Whereas **ts + tc** describe the time takes for a message to travel from Client Service to Cloud Server where server does some processes to serve the Client Service's request. If it only says **ts** or **tc** and does not have a returning **tx** it means that the process does not require a return from the Cloud Server, (this could mean that the return message from the Cloud Server is processed by another process).

6.2. Database Notation (SQL comparison):

- Ȝ for Select or search query,
- Ꭲ for insert new row,
- ∅ for remove row and
- m for modify row.

6.3. Algorithm Notation:

- MFSChar **- ::** indicates true path &
- **;;** indicates false path.
- Multiple of these gives a description of path needed to reach eg. **::;;::** = true false true, An example is **Z::** should be read Z algorithm, First = true.



Pseudo code	Cost	Times
A		
Start_Session()		
Connect_To_Server( );	c0	J
If(CONNECTED)	c1	1
Print(LOGIN_PAGE);	c2	1
Else	c3	1
Print(CONNECTION_ERROR_PAGE);	c4	1
Halt;	c5	1

Time O(J): Connect to Cloud server (Hand Shaking or some sort of security encryption process if needed). Connection will verify ID (ID is SPID, IP, and MAC). The ID will be encrypted; Cloud server will confirm ID as first step, if it is Valid. In case the client ID is valid, it will notify user by log-in display, accompanied by encryption process (example: a key exchange and a confirmation code (Daemon or process that is not visible to user)). Client service side generation of login reduces server process time, it also aids in prevention of undesired access. Else client will generate its own error page, **Halt** meant to end session and close program.

grs + **E**

True**::** { -> + Ȝloginpage + printLoginPage;
False**;;** { -> + ȜError page + printErrorPage + Halt;

T = O(ec) or O(ds): **p** depends on how many fields (or data inputs the user is needed to fill) are in the login.

© 2012 GSTF

B

```
Go_Log-in_Request()
        If( Log-in_Parameter_Check )                                           c0    p
                Send( Encrypt_1c( LOGIN_NAME, PASSWORD ) );                    c1t   ec1
                Server_Request_ID(( Decrypt_2s( Server_Request_1 ) );          c2    E
                Log-in_Response( SERVER_RESPONSE_1 );                          c3    C
```

Time = O(p). When the Client service user initiates the log-in request (by a button or some other method) log-in name and password must not be null. (This will reduce failed attempts also the Cloud server will check for null fields and mark the user as compromised if finds one. It will send the encrypted log-in name and password along with a session key, and then waits for the Cloud server response, and decrypts for next process.

Log-in_Response (SERVER_RESPONSE_x).

$\sum_{P0}^{Pn} p + \sum_{P0}^{Pn} ec1$ + ts + tc -> + **E**:: -> + tc + **C**::;

C

```
Log-in_Response( SERVER_RESPONSE_x )
        Decrypt_1s ( SERVER_RESPONSE_x )                                       c0    ds1
        If ( ACCEPTED_LOGIN ) // if the ID is valid to the server              c1    1
                On_Log-in_Response( DECRYPTED_SERVER_RESPONSE_x );             c2    D
        Else                                                                   c3    1
                Print ( LOGIN_PAGE );                                          c4    1
                Go_Log-in_Request( );                                          c5    1
```

T = O(ds): Decrypts the Cloud server's response, from the log-in message, that was sent. If the log-in is accepted it will process **On_Log-in_Response(RESPONSE).** If not accepted it will go back to the log-in page.

$\sum_{StartSResponse}^{EndSResponse} ds1$ + Accepted

True:: { -> **D;**

False:: { -> ȢError page + printErrorPage + **B**;

D

```
On_Log-in_Response( SERVER_RESPONSE_w )
        Server_Request_ID( Decrypt_2s( SERVER_RESPONSE_w ) );                  c0    E, ds2
                If ( ACCEPTED ) // Log-in Parameters are accepted              c1    1
                        Display_Session_Page( DECRYPTED_SERVER_RESPONSE );     c2    1
                Else                                                           c3    1
                        Go_Log-in_Request( );                                  c4    B
```

T = O(ds) or O(E) or O(B): Will first decrypt the Cloud server's response. From the server's response the Client service will determine if user is logged in or not, and if ID is valid. If accepted it will generate a session page. If not accepted will go back to the log-in page.

$\sum_{StartSResponse}^{EndSResponse} ds2$ + **E**:: + tc + Accepted

True:: { -> DsplayPage;

False;; { -> **B**;

T = O(ec): s are the other security measures.

Estimate T = O (1) to O(n): Will check to make sure the server is correct. If the key is correct will send the computer's SPID, IP, and, MAC. If the key is not correct it will go through a series of actions, locking and halting the session. Notifying the user of the session is compromised.

E

```
Server_Request_ID( SERVER_RESPONSE_z )
        Check_Server_Signature_Key ( SERVER_RESPONSE_z ) );                    c0    1
        If ( CORRECT_SERVER_KEY )                                              c1    1
                Send( Encrypt_2c ( Fetch( SPID, IP, MAC ) ) );                 c2+t  ec2
        Else                                                                   c3    1
                Session_Locked;                                                c4    1
                Request_Server_ID;                                             c5+t  ecy
                Message_User( SERVER_COMPROMISED );                            c6    1
                Record_Server_ID;                                              c7    1
                Send_to_Securities( Encrypt_4c( Server_ID, Client_ID,   SERV-  c8+t  ec4
ER_RESPONSE_DECRYPTED ) );
                Extera_Security;                                               c9    s
                Halt                                                           c10   1
```

Requesting and recording the Server ID. Send a message to Securities (Is another server or part of the network that will take action to threats). Finally halts until issue is taken care of. **Extra_Security** is something to the point that the program will not start again unless expressly authorized.

CheckSignature

True:: { + -> ȢID + $\sum_{StartID}^{EndID} ec2$ + ts;

OR

False::{ -> + c4 + ts + t6 + tc + ƮServerID + $\sum_{StartSID}^{EndSID} ec4 + \sum_{i=1}^{n} SecurityOperation\,[i]$ + Halt;

F

```
Request_Data( COMMAND_x )
        Send ( Encrypt_3c ( Generate_Data_Request( COMMAND_x ) ) );            c0+t  ec3
        Server_Request_ID ( ( Decrypt_2s( SERVER_RESPONSE_3 ) )                c1    ds2, E
        Generate_Session_Page ( DECRYPTED_SERVER_RESPONSE_4 );                 c2    1
```

T = O(ec): Sends an encrypted message to server the message is generated from the user's command. It will then process the server's request for the client user's ID. From the response the Session page will be displayed. Server reserves right to deny access to clients.(note: denial of access results in blacklist of client).

ȢDataReq + $\sum_{StartDataReq}^{EndDataReq} ec3$ + ts + tc -> $\sum_{StartResponse}^{EndResponse} ds2$ + PrintServerResponse;

G

```
On_Session_Exit( )
        Send( END_SESSION_CODE );                                              c0    1
        Halt;                                                                  c1    1
```

T= O(ec): It will send end session code to server to terminate the session; then halts program.

$\sum_{StartEndSession}^{EndEndSession} e$ + halt;

H

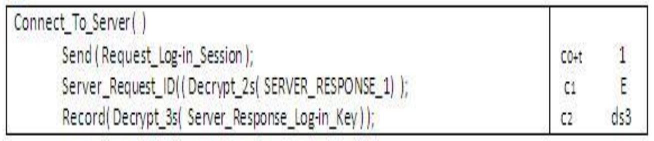
```
On_Command( )
        Request_Data( COMMAND_1 );                                             c0    F
```

T = O(F): It will send an encrypted request to server then Request_Data(COMMAND_MADE). Simple example for on-click or other GUI command.

See Request_Data(Command)

T = O(ds): Sends a request for login session to server. Server will request an id and client will decode the response and determine if the server is valid. Client will then record the login key.

J



```
Connect_To_Server( )
        Send ( Request_Log-in_Session );                          c0+t    1
        Server_Request_ID( ( Decrypt_2s( SERVER_RESPONSE_1) );    c1      E
        Record( Decrypt_3s( Server_Response_Log-in_Key ) );       c2      ds3
```

$$\text{GenerateRequestLogin} + \text{ts} + \text{tc} + \sum_{STARTServerResponse}^{ENDServerResponse} ds2 + \mathbf{E}{::} \text{->} + \text{ts} \text{->} \sum_{StartSResponse}^{EndSResponse} ds3 + \text{TDecryptedResponse};$$

Hence the conclusion for time analysis is that operation time is dependent on Encryption and Decryption algorithms used (**T = Θ(n)** best case and **T = O(n!)** worst case), if we ignore Encryption and decryption the runtime can be described as **T = O(1)** in most cases for the algorithms used, and **T = O(n)** in some processes such as **On_Log-in_Response()** which has a parameter check.

7. Conclusion

Given the ubiquity of wired and wireless high speed connections, there are no significant barriers to the performance of this solution. Data recharging service in a Cloud is a phenomenon of seamlessly transmitting secured data between ***n*** nodes after making sure that the user is authenticated. Intermittent connectivity of the Cloud, and is a main reason to use a good mix of the approaches to transmit data asynchronously. The research objective of this novel Client Service secured login in a Cloud is to improve users' safety and secured productivity by providing secured data access at anytime, anywhere and with consistent data transmission performance.

As described in the paper, though there are tremendous advantages in using cloud-based systems, there are yet many realistic problems which have to be solved. Cloud is a union of several hardware and software platforms. There are several hybrid technologies being utilized to provide the services in the cloud by many service providers. This paper sheds light on few of the several issues dealing with Cloud and specifically the security issues and research work done to find suited solutions to resolve security problems. It also encompasses a proposal of Cloud Security Architecture based on Layered Architecture approach. This approach is presented as Cloud Security Model and is named CSM 2.0, with expansion of three services with an addition of encrypted internal keys to find the source of request and destination, which needs more future enhancements and modifications with the incorporation of systems functional testing. CSM 2.0 has potential to be enhanced to achieve a favorable methodology to resolve the issues related to Cloud Security, reliable availability of the desired results from the effective use of Cloud's available resources.

References

[1] Atif. F. Mohammad, Emanuel. S. Grant; Cloud Computing, SaaS and SOA 3.0: A New Frontier. Cloud Computing and Virtualization 2010 International Conference, Singapore May 2010

[2] M. Rosen, B. Lublinsky, K.T. Smith, M. J. Balcer. Pulished. Applied SOA: Service-Oriented Architecture Design and Strategies. Wiley Publishing Ltd. 2008.

[3] R. Schreiner, U. Lang; Protection of complex distributed systems. Proceedings of the 2008 workshop on Middleware security. Pages 7-12. 2008.

[4] T. Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, 2005.

[5] http://www.amazon.ca/; Accessed on April 03, 2011.

[6] http://www.expedia.com; Accessed on April 10, 2011.

[7] Schneier Bruce, Secure System Engineering Methodology, http://www.counterpane.com/; Accessed on April 15, 2011

[8] H. P. Tipton and M. Krause, Information Security Management 4th edition, Auerbach, United States of America, 2000

[9] C. Dörner, V. Pipek, M. Weber, V. Wulf. End-user development: new challenges for service oriented architectures. in Proceedings of the 4th international workshop on End-user software engineering, pp. 71-75, May 2008

[10] F.P.J. Brooks. No silver bullet: essence and accidents of software engineering, IEEE Press, pp.10-19, 1987

[11] B. R. Kandukuri, V.R. Paturi, A. Rakshit. Cloud security issues.In: IEEE international conference on services computing, p.517–20. 2009

[12] V. Choudhary. Software as a service: implications for investment in software development. In: International conference on system sciences,p.209. 2007

[13] Glass, L. Robert. Facts and Fallacies of Software Engineering. Addison Wesley, 2006

[14] H. Liu, and D. Gluch. Conceptual modeling with the object-process methodology in software architecture. J. of Computing in Small Colleges, 19 (3), 10-21. 2004

[15] D. Dori, M. Choder.Conceptual Modeling in Systems Biology Fosters Empirical Findings: The mRNA Lifecycle. PLoS ONE 2(9): e872. 2007

[16] C. Hofmeister, R. Nord and D. Soni, Applied Software Architecture, Addison-Wesley, 2000

[17] Atif. F. Mohammad, Ronald. Marsh, Emanuel. S. Grant. Cloud Security Model using SOA 3.0 - CSM 1.0: A New Frontier. 2nd International conference on Cloud Computing and Virtualization 2011, Malaysia, April 2011

[18]http://www.pmddtc.state.gov/regulations_laws/itar_official.html; Accessed on April 19, 2011

[19] http://www.gpo.gov/bis/ear/ear_data.html; Accessed on April 19, 2011

[20] Wolski, A., "Embedding Data Recharging In Mobile Platforms", Real-Time & Embedded Computing Conference (RTEC'01), November, 2001

Authors

Emanuel S. Grant received a B.Sc. from the University of the West Indies, MCS from Florida Atlantic University, and a Ph.D. from Colorado State University, all in Computer Science. Since 2008, he is an Associate Professor in the Department of Computer Science at the University of North Dakota, USA, where he started as an Assistant Professor in 2002. His research interests are in software development methodologies, formal specification techniques, domain-specific modeling languages, and model driven software development.

He is an adjunct professor at the Holy Angel University, Philippines, where he is conducting research on software engineering teaching with collaborators from HELP University College, Malaysia; III-Hyderabad, India; Rochester Institute of Technology, Baylor University, Montclair State University, and University of North Carolina Wilmington of the USA. Emanuel is a member of the Association for Computing Machinery (ACM), Upsilon Pi Epsilon (UPE), and the Institute of Electrical and Electronics Engineers (IEEE).

Atif has more than sixteen (16) years experience of Software Engineering, professional Business Systems Analysis, Design & application development and staff management for diversified business and educational organizations. These constitute several commercial and academic institutions, oil, automobile distribution, couriers and garments manufacturing establishments. I have built systems using various application development tools and relational database management systems (RDBMS). Applications developed ranged from Personnel & Payroll, Sales, Purchase, Daily Courier Operations management, Quality Assurance & Inventory control of various types.

© 2012 GSTF