Journal of Quality and Technology Management
Volume IX, Issue II, December 2013, Page 107–136

REQUIREMENTS ANALYSIS ALGEBRA: DESIGNING QUALITY SOFTWARE

A. Nasir, A. Shah & M.U.G. Khan
Department of Computer Science and Engineering,
University of Engineering and Technology, Lahore, Pakistan

ABSTRACT

For software systems to be of good quality and acceptable to users, it is pre-requisite that the software developed so far must conform to user's needs at its best. Moreover, quality requirements are essential for a software system to be developed. The software must comply with users' requirements either functional and/or non-functional. The process of gathering and analyzing essential and complete user requirements is challenging one. As development of cloud applications is concerned, the requirements gathering and analysis process seems to be more complex due to crucial nature of user requirements particularly encompassing additional security requirements. Key challenge in cloud computing is data security so securities concerns are supposed to be the most essential factor in the domain of cloud computing applications. Such challenge demands a formally secure System Development Life Cycle (SDLC). Our research study proposes requirement analysis algebra that may provide a formal basis to requirement engineering and consequently may help the analyst in analyzing both functional and security requirements of cloud applications to be developed. The application of algebra proposed may enhance understanding of system requirements phase and improve the security requirements significantly.

Keywords: *Requirement Engineering, Requirement Algebra, Performance Monitoring, Cloud Application*

1) INTRODUCTION

Key challenge in cloud computing is data security so securities concerns are supposed to be the most essential factor in the domain of cloud computing applications. Such challenge demands a formally secure System Development Life Cycle (SDLC). A mathematical model to incorporate security requirements during development life cycle of a cloud computing application is introduced in our research study. To provide complete guideline to handle and incorporate security

requirements during life span of cloud computing environment, the Requirement Algebra (RA) as a tool is introduced and then used. Requirement Algebra (RA) can also be used for all such application domains where security is curious. For better understanding of the RA of functional and security requirements during analysis, a practical example of E-Legal system has been considered. Its detail is given in section 3.

The process of requirements gathering and analysis during software development is an important aspect and it's a difficult and critical task (Playle et al., 1996). The dependency of quality software on requirement gathering and their analysis cannot be under estimated (Microsoft, 2003; Holtzblatt et al., 1995). For cloud computing applications in which security is a major concern, the difficulty level of software application development is two fold because, gathering of functional and security requirements and their analysis in these applications is a time-consuming and pain-taking (Paolo et al.,2004). Identifying both functional and security related requirements and to find out inter and intra dependency relationships is a complex task (Mostert et al., 1995). The ever changing user requirements and security issues of the dynamic applications demands mathematical modeling techniques to be introduced for software design.

There exists, intra-relationships within the two sets of requirements, i.e., set of functional and set of security requirements and the inter-relationships between these two sets of requirements. The two sets of requirements are referred to as the set of operands of the proposed requirement analysis algebra. A set of operations/operators are defined as "operate on the set of operands". E-Legal system proposed in section-3 not only covers the homogenous but includes the heterogeneous domains such as Online-Banking collaboration with E-Legal and Prison Management consequently its complexity is twofold. Like any other internet based computing, major concern about Cloud Computing is its security (Harauz et al., 2009, Furht et al., 2010).

Mathematical modeling is speculated to be the best way to elaborate the security requirements (i.e. formalism) which helps to truly depict the requirements during analysis and will be used during design of system. To address this concern, interpretation of the operations and operands of the functional and security requirements of an E-Legal system in the cloud computing environment has been presented in the reaming part of

the paper. The proposed Requirement Analysis Algebra presented so far may serve as foundation for the secured design of cloud applications. It can be helpful in analyzing applications in the analysis phase of the proposed methodology. The salient feature of the algebra is that it built a table of user verse their opted requirements using the *Opt* operator (see Section 4.3.4). The remainder of the paper is organized as follows. In Section 2, we give the related work. In Section 3, we give a running example and its details. This example is used to elaborate working of the proposed algebra. The proposed requirement analysis algebra is presented in Section 4. Whereas, implementation of the *RAA* is provided in Section 5. In Section 6, we give concluding remarks and future directions of this work.

2) LITERATURE REVIEW

According to National Institute of Standards and Technology (NIST), Cloud Computing is a shared pool of virtual resources such as hardware and software entities(like servers, network and software application services) to be provided on request as and when ever required that is scalable and managed with nominal effort (NIST, 2009). Different types of cloud computing deployments include "Public" (open to any customer for usage), "Private" (specifically for an exclusive use to limited type of the users, like a Virtual Private Network) and "Hybrid" i.e. combination of both public and private, (Ran, L.; et al., 2010).

Cloud computing is a new era in the computing world and getting popularity due to its various advantages. It is playing a vital role these days in various application domains enabling organizations to enhance their services by utilizing its economical power (Zheng, 2010). Cloud Computing is economical because of utilization of the idle I.T. resources through improving utilization rate and reducing power consumption (Berl et al., 2010).

Internet based computing services such as Cloud is the most incredibly and dynamically growing computer technologies in this modern era. (Leavitt, 2009) reported that the cloud adoption rate has tremendously accelerated. The number of users accessing the Cloud applications is growing exponentially all over the world (Buyya et al., 2009). One of the example of cloud computing is G-mail. Cloud computing is a scale-able pool of virtual resources (hardware, software and data etc.), promising

users to access these resources at anytime, anywhere through internet. For most of the users it is not different from the web services, however, technically both services have different attributes such as scalability, flexibility and resource pooling and these are the key differences of the Cloud (Dan Svantesson et al., 2010).

2.1) Cloud Computing Services

Cloud computing is playing an important role these days in various fields enabling the Organizations to improve their service delivery by utilizing its economical power (Raj et al.,2011). Cloud computing paradigm is a transit from the single customer approach towards scalable multi-users on multi-platform computing services through Internet (King, 2008). Software as a Service (SaaS) are software applications that are provided as a service to the users. Financial, Inventory, Supply chain application, HR and CRM are most common categories of Software as a Service; the most prominent vendor in this category is Salesforce.com. The payment of the software is on use basis and does not require any capital investment by the user (Armbrust et al., 2010).

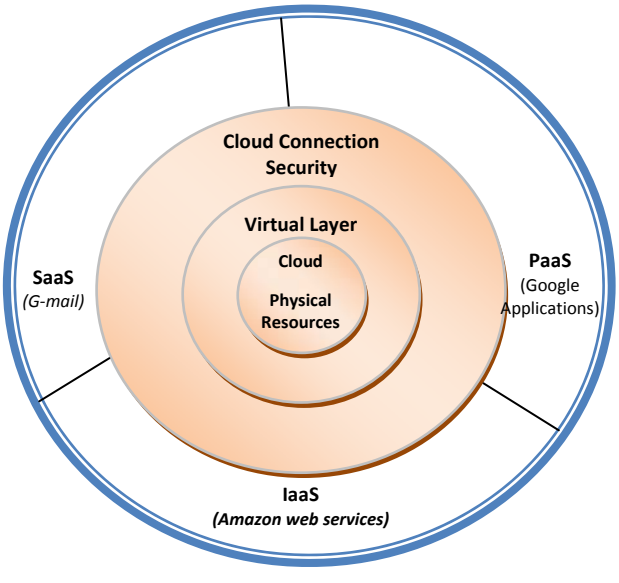

Figure 1: Main Services of Cloud Computing

A SaaS consisting of a single workload is most cost effective for the provider to manage and is therefore most economical model. SaaS suppliers offer the services accessible by using web browser, which does

not require any installation by the end-user premises. Common examples include Google's Gmail, Google Search, Microsoft's Business Online Productivity Suite, Microsoft Customer Relationship Management (CRM), Microsoft Office Live, salesforce.com on demand applications etc. In contrast to IaaS, SaaS provide solutions of the specific problem that includes software applications hosted as service, provided to customer. As in the perspective of SaaS, one program is sufficient to serve millions of users with the ultimate benefit of cost saving (Jaeger et al., 2008). Approximately, 50% of the internet users are using cloud services in one-way or the other (Yunis, 2009).

Cloud computing extends cross-organizational and regional boundaries (Sasaki, 2011). Since cloud services are available globally across the board on multi-platforms. Therefore, security and privacy issues of cloud computing have to be resolved, to utilize full potential of this new emerging platform of the future (Glott et al., 2011).

Due to peculiar characteristics of cloud computing, there are many problems in this new computing environment and security concern is one of the major issues of the users and service providers (Armbrust et al., 2010). Security requirements may affect the working of software (Rashid et al., 2003). Privacy should be integrated at every stage of software development instead of together the screw at some later stage (Ran, L.; et al., 2010). It has also been observed (Giorgini et al., 2005), that Security requirements often conflict with each other, as well as with other requirements. Therefore, it is important to understand the intra-relationship of security and functional requirements separately, as well as the inter-relationship between security and functional requirements.

It has been emphasized revision of existing development methodologies to ensure security of cloud application throughout the development life-cycle (Javier et al., 2008). The concept of cloud computing cannot be successfully implemented, until and unless a secured mechanism is introduced. It is therefore justifiable to investigate and propose methodologies to ensure reliability and security of cloud computing, which requires in-depth study and analysis that how security to be considered for modelling and development; from the initial phase to obtain secure cloud systems. The formal methods are disciplined techniques so its usage is helpful for specification, development and verification of software (Chen et al., 2004). An example of such successful

implementation of formal methods to ensure software security is type checking (Chen et al., 2002), is an integral feature of modern programming languages like Java and C#. The formal methods best suit to precisely specify such requirements to be addressed during architecture and design stages (George et al., 2003). According to (Butler et al., 1995) mathematical (or formal) methods and notations are used to precisely define the behaviour of the software, consistent with its requirements, and to model its properties and attributes.

Since formalism is considered the best way to maintain maximum level of accuracy in terms of security, therefore Requirements Analysis Algebra proposed may serve as secured foundation for the whole future IT world.

3) E-LEGAL SYSTEM

In order to understand requirement algebra of the functional and security requirements during analysis of a cloud application development, real life example of E-Legal system is being discussed as a representative of all application domains. An E-Legal system is aggregation of various domains, because of its interaction with other application such as Prison Management, On-line Banking, E-Govt., Solicitor Management, Forensic Lab system etc. That is why; it best suits to elaborate our research work.

However, due to peculiar characteristics of cloud computing, its utilization varies from domain to domain. Although, electronic legal system is beneficial in dispensation of fast and transparent justice to litigants as well as legal fraternity but at the same time its utilization in legal domain is still questionable due to security and privacy of the information and sharing of such information with the other stakeholders of the legal system such as Government Prosecutors, Advocates, law enforcement agencies, Forensic Labs., and the litigant public etc.

The electronic legal system in a court of law is part of judicial process and includes different sets of electronic functions and technology gadgets to support the Court's working. Different working modules are shown in an electronic legal system (see Figure 2) for example Electronic Case Filing, Scrutiny, Case Fixation, Scheduling, Court Proceedings, Workflow, Court's decision and judgment etc.,

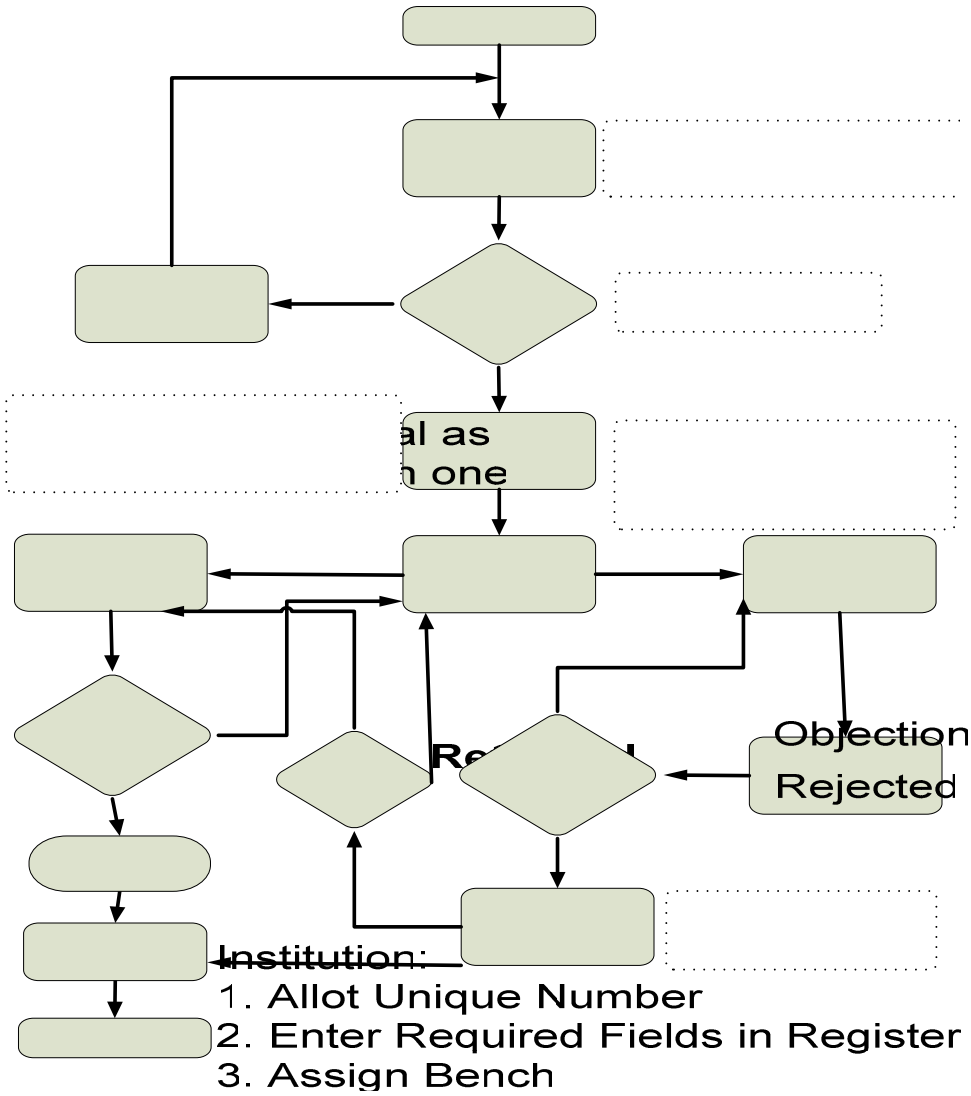

Figure 2: Abstract view of E_Legal System Modules

Case Filing is the first and foremost important step of the judicial process, which is the foundation of a legal proposition. Maximum contents (e.g. attaching copies of documents such as FIR, Medical report, Fardmalkiat, stamps, grievance and prayer of the case) of the case are added at this stage and all these types of contents fall in different types of categories. So, at the one end the data diversity is high at this stage of the legal system and at the other hand its volume is very high. A number of steps/procedures (as given in Figure 3) are involved at the time of filing a case. Moreover, payment of the court fee depicts financial aspect of the case. Hence security level required during filling of a case is very curious. Electronic

filing lets people get more of their work done with their PCs to send and receive documents, pay filing fees, intimate other parties, receive court notices, and retrieve court information etc. As shown in Figure 3, during this process, documents and other court information is transmitted to the court through an electronic medium.

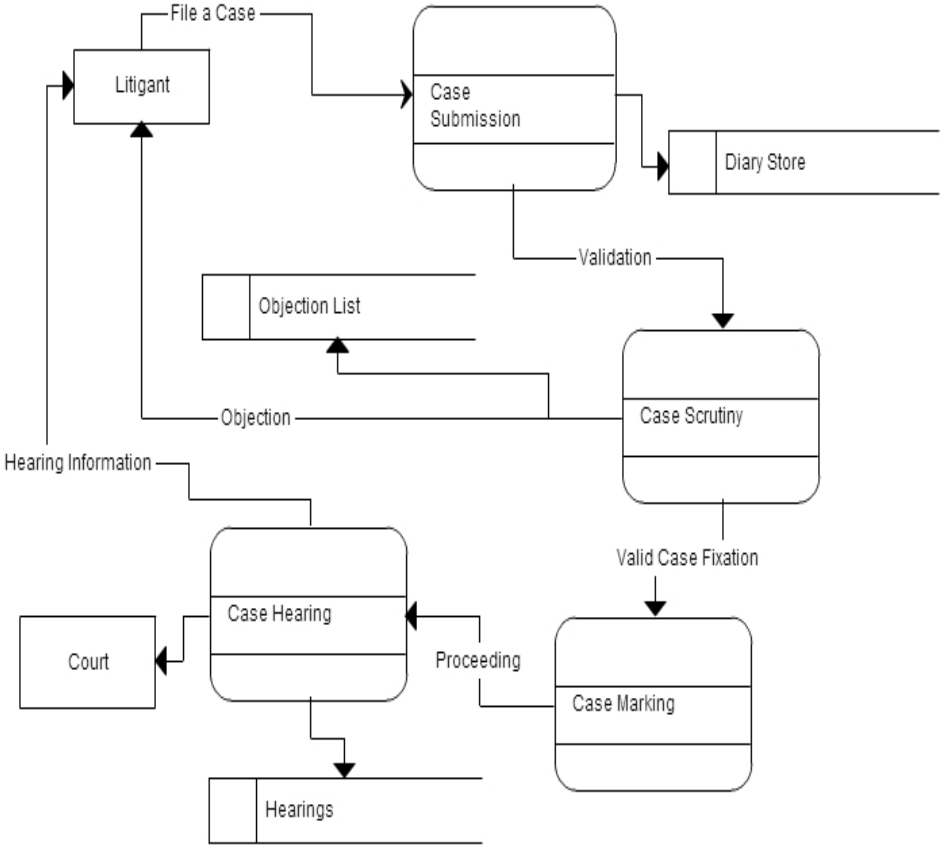

Figure 3: DFD of E-Filling

During Case filing, Court fee is also paid and it also includes the case diary process, scrutiny and allocation of Bench/Judge. After a case is diarized, it is scrutinized against case law and other relevant court rules, as provided in the check-list of objections. If the case has no objection then case is considered as fit for hearing and a case number is assigned. The System marks the case for hearing to a bench/judge depending on the criteria set after checking the roster of available Judges. The case is also allocated to a Bench on the basis of category or specialty (e.g. Civil or Criminal) of the Judge and even on the basis of value involved or

sentence by lower court, if applicable (such as death life imprisonment etc.).

4) REQUIREMENT ANALYSIS ALGEBRA (RAA)

In software engineering, requirements are categorized broadly into two types, i.e., functional requirements and non-functional requirements. But in the cloud application domain and some other application domains (e.g., as web applications), during the development process main focus is on security requirements of the applications and these requirements may be considered as a part of functional requirements. But here we consider functional requirements and security requirements separately. By separating these two types of requirements, we can concentrate and analyze security requirements, functional requirements and their dependencies. To study and analyze both types of these requirements, we propose *Requirement Analysis Algebra* (RAA). This algebra can play an important role during analysis of all types of applications, particularly for those applications in which security is a crucial factor. Before proposing RAA we categorize requirements of an application and give their properties in the next section.

4.1) Atomic and Composite Requirements

We categorize requirements in two types/categories, i.e., *Atomic* requirement and *Composite* requirement. They are categorized on the constituent basis of requirements. Assume that the set, $F = \{F_1, F_2, F_3 \ldots\ldots\ldots\ldots Fn\}$, is the set of functional requirements of an application/software. Each subset of the set *F* may be *Atomic* or *Composite* requirement. We define a requirement to be an *Atomic Requirement* (AR), if it is not further divisible into a sub-requirement(s). In the case study given in Section 3, *F34* (see details in Table 5.1), i.e., *Issuance of the unique case no* is an *Atomic* requirement. We refer to a requirement as the C*omposite* Requirement (CR), if it is further divisible into one or more *atomic* and/or *composite* requirement(s). For example, in the case study F_8, i.e., *Submission of respondent information* as a whole is a composite requirement. Because F_8 consists of *Respondent's Name(s)*, *Address(s)* and *Contact(s)* etc. It is to be noted that in our *RAA,* all requirements are in *Atomic* form and *Composite* requirements have been also decomposed in to *Atomic.*

4.2) Requirement Types

In this section, we classify software requirements of an application based on their functionality. This classification is done into two types, i.e., *functional requirements* and *security requirements*. The set of functional requirements is denoted by $F = \{F_1, F_2, \ldots..F_n\}$. Similarly, the set S is the set of *security requirements*, and it is denoted by $S = \{S_1, S_2, \ldots, S_m\}$. The set R denotes both types of requirements (i.e., functional and security requirements) of a software/application, and they are written as, $R=\{F \cup S\}$.

These two sets that are usually non-empty sets, i.e., S and F are operands of the proposed *RAA*. The *intra* and *inter* operators of the proposed algebra are defined in next Section.

4.3) Operators

In this section, we define operators of the proposed algebra. The proposed operators operate on the operands that have already been defined earlier in Section 4.1. These operators are formally defined in the next sections.

4.3.1) Need Operator

We define *Need* operator to develop the relationship between the two sets F and S. If a security requirement, S_j, is a mandatory requirement for the functional requirement, F_i, to make it secure, then we say that the security requirement, S_j, is a *Need* (N_d) of the F_i requirement and it is denoted as; N_d: $F \rightarrow S$. The N_d operator is applied on the elements of the set F and it returns elements of the set S, i.e., $N_d(F_i) = Select\ (S_j);\ (\ n: 1 \leq i \leq n\ )$ *and* $(\ m: 1 \leq j \leq m)$ and $0 \leq |N_d| \geq n$.

In other words, this operator finds security requirements of a given functional requirement. In our case study given in section 3, the functional requirement, F_{22}, i.e., *payment of court fee,* $N_d(F_{22}) = \{S_{13}, S_{14}, S_{15}, S_{16}\}$, and using this operator N_d returns the element S_{13} *Checking of availability of funds in the relevant account,* as the security requirement and the elements S_{14}, $S_{15}$and S_{16} are also returned as security requirements of the functional requirement F_{22}.

4.3.2) Vital Operator

We define *Vital* operator to find out the vitality of security requirements for different functional requirements. If a security requirement, say S_j, is required and it is mandatory for the functional requirement(s) F_i, then *Vital* (ŏ) of S_j is F_i. We denote it as; ŏ(S) = F. The ŏ operator is applied to elements of the set *S*, and it returns elements of the set *F*, i.e. ŏ(S_j) = *Select* (F_i); ($n: 1 \leq j \geq m$ $i: 1 \leq j \geq n$).Whereas1 < | ŏ (S_j) | > n

In our case study the set S_9, i.e., *On-line verification of advocate from Bar Associations* ŏ (S_9) = {*F1, F13, F19, F21*} that means S_9will vital for F_1, F_{13}, F_{19} and F_{21} to make them secure.

4.3.3) Dependent Operator

To find out *dependency* among different elements of the set, *R*, of requirements, we use the *dependent* operator and we denote it as Ď. Whereas, set *R* is total requirement i.e. functional and security such that

$R = F \cup S$ and $F \cap S = \phi$

If a requirement has pre-requisite(s) that may be functional or security requirement(s), then the operator among them is referred as *dependent.* This system (F, Ď) represent dependency among different functional requirements. If a functional requirement, F_j, is dependent upon F_i, then F_i is pre-requisite and should act before the action of F_j. We will denote it as;

Ď:F → F; ∀ $F_i, F_j \in F$, Ď(F_i) = F_j, $i \# j$. For example, in the case study (see

section 3) F_3, i.e., *Payment of E.F. Charges* dependent upon, F_2, *Selection of Electronic Facilities (EF)*. Similarly, dependency among security

requirements is represented as;.Ď:$S \rightarrow S$; ∀$S_m, S_n \in S$, Ď(S_m) = $S_n \wedge m \# n$.

4.3.4) Opted Operators

To capture the majority-honored requirements, we propose *Opted* operator. This operator is denoted by $\hat{O}_{pt}$, and it returns number of users who have opted for a given functional requirements. The $\hat{O}_{pt}$ operator can help the analysts in determining the percentage of users for each requirement of an application, who have opted the requirement out of the total number of requirements of the application.

For instance, U_1, U_2, U_3,....U_k are the users and they have opted functional requirements using the operator $\hat{O}_{pt}$. Results of the operator $\hat{O}_{pt}$ got this set of users are recorded in Table 1. This table is initially built during Analysis Phase, and it can be used during and after development to monitor functional requirement of an application. This monitoring can be helpful to the system administrator in deciding the future of each functional requirement/service.

Table 1: Monitoring of functional requirement

User	**F_1**	**F_2**	**F_3**	**…>**	**F_{n-2}**	**F_{n-1}**	**F_n**
U_1	$\hat{O}_{pt}$	$\hat{O}_{pt}$			$\hat{O}_{pt}$		
U_2	$\hat{O}_{pt}$				$\hat{O}_{pt}$		
U_3	$\hat{O}_{pt}$						
.	..	…		..	..	..	..
U_k	$\hat{O}_{pt}$	$\hat{O}_{pt}$					
Enumerate	4	2			2		

This is a dynamic type table, and as said earlier it is built during the analysis, and then it can be updated during the subsequent phase. It is also updated regularly when the application is operational. Note that this table becomes permanent part of the cloud application after its development. Based on these monitoring results of the table, the application can be tune-up to get better performance of the application after its development. This can save computational cost and also reduce the over-heads by temporarily disabling the functions, which have value of $\hat{O}_{pt}$ less than threshold value. Thereafter, when the value of the

operator, $\mathring{O}_{pt}$, for a certain function increases than its threshold value, then the function can be enabled. In Figure 4, we have shown a set of requirements/demands of different user's, they are given as follows: $\mathring{O}_{pt}(U1)$, $\mathring{O}_{pt}(U2)$, $\mathring{O}_{pt}(U3)$……$\mathring{O}_{pt}(uk)$.

$$\mathring{O}_{pt}(U1) = \{F_1, F_2, Fn_{-2}\}$$

$$\mathring{O}_{pt}(U2) = \{F_1, F_{n-2}\}$$

$$\mathring{O}_{pt}(U3) = \{F_1\}$$

$$\mathring{O}_{pt}(Uk) = \{F_1, F_2\}$$

Figure 4. User's Requirements Monitoring

4.3.5) Union and Intersection Operations

Now we define the set of operations/operators *Union* and *Intersection on* user requirements to assess their usefulness. This can be helpful to the software engineer/analyst.

i) Union Operator

Requirements of application are gathered from multiple and different sources and hence are prone to be duplicated. The *union* (*U)* operator filters out the redundant requirements, it operates on set requirements of different users (as shown in Table 1), and gives a unique set of

requirements of clients i.e., $F = \{F_1, F_2, F_{3....} F_{n-2}, F_{n-1}, F_n\}$. $\forall F_i \epsilon F$. If total

number of users of an application is k, then $\mathring{O}pt$ *(Requirement) = Enumerate/k*

$\Rightarrow \mathring{O}_{pt}(F_1) = 4/k'\ \mathring{O}_{pt}(F_2) = 2/k'\ \mathring{O}_{pt}(F_{n-2}) = 3/k$ *and so on* $\forall F_i \epsilon F$

We develop only those requirements which have been $\mathring{O}pt$ by the numbers of users greater than the pre-determined threshold. The threshold will is calculated by the analyst, using percentage of users.

Finally, we obtain the set of candidate requirements to be analyzed by the analyst for the software development.

ii) Intersection Function

The *Intersection* operator returns the most important and popular requirements of an application among its users. It provides the rating of requirements as well as determines commonality. The commonality helps to develop such requirements which are used by large number of users, and it helps in the rating application's requirements highest in demand. The *Intersection (∩)* operator operates on the set of requirements. It is observed that the operator U and ∩ differ, because of their objective.

4.4) Rules for operator *Need* (N_d)

In Appendix-I and Appendix-II, we give functional requirements' dependency and their security *Need* (N_d) and security features for various functions, respectively, of our case study/running example given in Section 3. We have classified the security requirements of various functions such as *High*, *Medium* and *Low* on the basis of their vulnerabilities. Similarly, in Appendix-III, the input and output are categorized based on their importance and vulnerabilities as *Open* (O), *Close* (*C*), *Secret* (*S*) and *Top Secret* (*T*) and as *Open* (*Op*), *Close* (*Cl*), *Secret* (*Se*) and *Top Secret* (*To*), respectively.

We assign the attributes as O*pen, Close, Secret* and *Top secret* to an input and output if it accessible to public, if it is only accessible to user after certain timeframe, e.g., court decision, if it is accessible to only limited number of users and it is accessible to only restricted users after certain timeframe, respectively. The categories of security Needs operator, N_d, are proposed, i.e., *Low*, *Medium* and *High* to assign the applications in which securities are a crucial need. We are actively working to propose rules for boundary cases such as when there is no input or output of a function. In the following section, we give the rules to apply the N_d operator.

4.4.1) Rule for Single Input and Single Output

Input * Output = [O, C, S, T] * [Op, Cl, Se, To]

={(O, Op), (O, Cl),(O, Se) ,(O, To) ,(C, Op) ,(C, Cl),(S, Se) ,(C, To) ,(S, Op),(S, Cl), (C, Se) ,(C, To) , (T, Op), (T, Cl), (T, Se) , (T, To)

i) If (O, Op) v (O, Cl) ∈ (Input * Output) => Security "*Low*"

ii) If (O, Se) v (C, Op) v (T, Op) v (S, Op) v (C, Cl) ∈ (Input * Output) => Security "*Medium*"

iii) If (O, To) v (S, Se) v (S, Cl) v (S, To) v (C, Se) v (C, To) v (T, Se) v (T, Cl) v (T, To) ∈ (Input * Output) => Security "*High*"

4.4.2) Rule for Double Input and Single Output

Input * Input * Output = [O, C, S, T] * [O, C, S, T] * [Op, Cl, Se, To]
={(O, O, Op), (O, O, Cl),(O, O, Se),(O, O, To), (O, C, Op), (O, C, Cl),(O, C, Se),(O, C, To),
(O, S, Op), (O, S, Cl),(O, S, Se),(O, S, To), (O, T, Op), (O, T, Cl),(O, T, Se),(O, T, To),
(C, O, Op), (C, O, Cl),(C, O, Se),(C, O, To), (C, C, Op), (C, C, Cl),(C, C, Se),(C, C, To),
(C, S, Op), (C, S, Cl),(C, S, Se),(C, S, To), (C, T, Op), (C, T, Cl),(C, T, Se),(C, T, To),
(S, O, Op), (S, O, Cl),(S, O, Se),(S, O, To), (S, C, Op), (S, C, Cl),(S, C, Se),(S, C, To),
(S, S, Op), (S, S, Cl),(S, S, Se),(S, S, To), (S, T, Op), (S, T, Cl),(S, T, Se),(S, T, To),
(T, O, Op), (T, O, Cl),(T, O, Se),(T, O, To), (T, C, Op), (T, C, Cl),(T, C, Se),(T, C, To),
(T, S, Op), (T, S, Cl),(T, S, Se),(T, S, To), (T, T, Op), (T, T, Cl),(T, T, Se),(T, T, To)

i) If (O, S, Op) v (O, C, Op) ∈ (Input * Input * Output) => Security "*Low*"
ii) If (O, S, Se) v (O, C, Se) v (O, S, Cl) v (O, T, Op) v (S, C, Op) v (C, T, Op) v (C,T, Cl) ∈ (Input * Input * Output) => Security "*Medium*"
iii) If (O, S, To) v (O, C, To) v (O, T, Se) v (O, T, Cl) v (O, T, To) v (S, C, Se) v (S, C, Cl) v (S, C, To) (C, T, Se) v (C, T, To) ∈ (Input * Input * Output) => Security "*High*"

4.4.3) Rule for Double Input and Double Output

Input * Input * Output * Output =[O, C, S, T]*[O, C, S, T]*[Op, Cl, Se, To]*[Op, Cl, Se, To]
={(O, O, Op, Op), (O, O, Op, Cl), (O, O, Op, Se), (O, O, Op, To), (O, O, Cl, Op), (O, O, Cl, Cl), (O, O, Cl, Se), (O, O, Cl,To)…… (T, T, To, To)

i) If (O, O, Op, Op) v (O, O, Op, Cl) v (O, O, Cl, Op)... ∈ (Input * Input * Output* Output) => Security "*Low*"
ii) (O, O, Op, Se) v (O, O, Cl, Cl)… ∈ (Input * Input * Output* Output) => Security "*Medium*"

iii) If (O, O, Op, To) v (O, O, Cl, Se) v (O, O, Cl, To) v (T, T, To, To)…
∈ (Input * Input * Output* Output => Security "*High*"

Similarly, we can extend these rules for multiple options of composite inputs and output. Moreover, we draw a conclusion that security such as *High*, *Average* and *Low* is required on the basis of types of input(s) and output(s) in Table 2. In our future work, we will also consider Operation, Consequences and Probability.

Table 2: Security required on the basis of Types of Input(s) and Output(s)

Input * Output (Data/Information Type)	**Security**
(O, To) v (S, Se) v (S, Cl) v (S, To) v (C, Se) v (C, To) v (T, Se) v (T, Cl) v (T, To)	*High*
(O, Se) v (C, Op) v (T, Op) v (S, Op) v (C, Cl)	*Medium*
(O, Op) v (O, Cl)	*Low*

5) IMPLEMENTATION OF *RAA* IN CASE STUDY

As, we have taken case study of E-legal system (for details please see Section 3), for better understanding of the *RAA* of the functional and security requirements during analysis. The main stake holders of the E-legal system are Advocate, Litigants, Court's staff, and judges. For better understanding of the Algebra and its use in the requirement analysis, we have assumed functional and security requirements of E-Legal system on the basis of our experience and discussion with main stake holders of the domain, which are illustrated in Table 3. $N_d(Fi)$.

$N_d(F_i)$ = *Select* $\{S_j: 1 \leq j \geq 16\}$.whereas $1 \leq j \geq 36$

Table 3: Functional Requirements of E-Filling

Depend	Function Requirements		**Security**
Input	F_1	Provide user's information	S_1, S_2, S_3, S_9
	F_2	Selection of Electronic Facilities (E.F.)	
F_2	F_3	Payment of E.F. Charges	S_{13}, S_{14}, S_{15}, S_{16}
F_1	F_4	Generate User-id	S_{11}

We have provided complete *RAA* of functional requirements of the in E-Legal system in Figure 6.

$$N_d\ (F_1)=\{S_1,S_2,S_3,S_9\}$$
$$\check{D}\ (F_3)=F_2$$
$$N_d\ (F_2)=\{S_{13},\ S_{14},\ S_{15},\ S_{16}\}$$
$$\check{D}\ (F_4)=F_1$$
$$N_d(F_4)=S_{11}$$
$$\check{D}(F_5)=F_4$$
$$N_d\ (F_5)=\{S_4,S_5.S_6,S_{11},S_{12}\}$$
$$\check{D}\ (F_6)=F_2$$
$$N_d\ (F_6)=S_8$$
$$\check{D}(F_7)=F_2$$
$$N_d\ (F_7)=S_8$$
$$\check{D}\ (F_8)=F_7$$
$$N_d\ (F_8)=S_8$$
$$\check{D}\ (F_{12})=F_4$$
$$N_d\ (F_{12})=S_8$$
$$\check{D}(F_{13})=F_{12}$$
$$N_d\ (F_{13})=S_9$$
$$\check{D}\ (F_{14})=F_4,F_7$$
$$\check{D}\ (F_{15})=\{F_7,F_{14}\}$$
$$N_d\ (F_{15})=\{S_{10},\ S_{11},\ S_{12},\ S_{13}\}$$
$$\check{D}\ (F_{17})=\{F_7,\ F_8\}$$
$$N_d\ (F_{17})=S_8$$
$$\check{D}\ (F_{18})=\{F_7,F_{13}\}$$
$$N_d\ (F_{18})=S_8$$
$$\check{D}\ (F_{19})=F_{13}$$
$$N_d\ (F_{19})=S_9$$
$$\check{D}\ (F_{20})=F_{13}$$
$$N_d\ (F_{20})=S_8$$
$$\check{D}\ (F_{21})=\{F_7,F_{13}\}$$
$$N_d\ (F_{21})=\{S_8,\ S_9\}$$
$$\check{D}\ (F_{22})=(F_2,F_{11})$$
$$N_d\ (F_{22})=\{S_{13},\ S_{14},\ S_{15},\ S_{16}\}$$
$$\check{D}\ (F_{23})=F_{22}$$
$$N_d\ (F_{23})=S_{13}$$
$$\check{D}\ (F_{24})=F_{23}$$
$$N_d\ (F_{24})=S_{13}$$
$$\check{D}\ (F_{25})=F_{15}$$

$$\check{D}(F_{26}) = F_{11}$$
$$\check{D}(F_{27}) = F_{25}$$
$$\check{D}(F_{28}) = F_{26}$$
$$\check{D}(F_{29}) = F_{15}$$
$$\check{D}(F_{30}) = \{F_7, F_{12}\}$$
$$\check{D}(F_{31}) = \{F_7, F_{12}\}$$
$$\check{D}(F_{32}) = F_{27}$$
$$\check{D}(F_{33}) = \{F_5, F_6\}$$
$$\check{D}(F_{34}) = F_{28}$$
$$\check{D}(F_{35}) = F_{34}$$
$$\check{D}(F_{36}) = F_{35}$$

Figure 5: Requirement Algebra of Functional Requirements

Figure 5, shows that F_1 is not reliant upon any requirement, however security S_1, S_2, S_3, S_9 are N_d for it.F_3 is $\check{D}$ upon F_2 and S_{13}, S_{14}, S_{15}, S_{16} is N_d for it. F_4 is $\check{D}$ upon F_1 and security S_{11} is N_d for it. F_5 is $\check{D}$ upon F_4, however S_4, S_5, S_6, S_{12} are N_d for it. F_6 is $\check{D}$ upon F_2; however security S_8 is must for it. F_7 is $\check{D}$ upon F_2, however it N_d is S_8. F_8 is $\check{D}$ upon F_7, and S_8 is N_d for it. F_{12} is $\check{D}$ upon F_4, and S_8 is N_d for it. F_{13} is $\check{D}$ upon F_{12}; however security S_9 is N_d for it. F_{14} is $\check{D}$ upon F_4 and F_7.F_{15} is $\check{D}$ upon F_7 and F_{14}, whereas security S_{10}, S_{11}, S_{12}, S_{13} is N_d for it. F_{17} is $\check{D}$ upon F_7 and F_8 and S_8 is N_d for it. F_{18} is $\check{D}$ upon F_7 and F_{13}, whereas S_8 is N_d for it. F_{19} is dependent upon F_{13} and S_9 is N_d for it. F_{20} is $\check{D}$ upon F_7 and S_8 is N_d for it. F_{21} is $\check{D}$ upon F_7, F_{13}, however security S_8,S_9 is N_d for it. F_{22} is dependent upon F_2, F_{11} however security S_{13}, S_{14}, S_{15},S_{16} are N_d for it. F_{23} is $\check{D}$ upon F_{22} however security S_{13} is N_d for it. F_{24} is $\check{D}$ upon F_{23} and S_{13} is N_d for it. F_{25} is $\check{D}$ upon F_{15}, F_{26} is $\check{D}$ upon F_{11}, F_{27} is $\check{D}$ upon F_{25}, $F_{28}\check{D}$ upon F_{26}, $F_{29}\check{D}$ upon F_{15}, $F_{30}\check{D}$ upon $\{F_7,F_{12}\}$, $F_{31}\check{D}$ upon $\{F_7,F_{12}\}$, $F_{32}\check{D}$ upon F_{27}, $F_{33}\check{D}$ upon $\{F_5,F_6\}$, $F_{34}\check{D}$ upon F_{28}, $F_{35}\check{D}$ upon F_{34} and $F_{36}\check{D}$ upon F_{35}.

We observe from Table 3, that some functions like F_1, F_2, F_3 *and* F_4 are totally independent. However, others including F_5 and F_6 *etc.*, are dependent; e.g., F_5 is dependent on F_1 that means that F_1 is pre-requisite to be executed before F_5. Moreover, transitive property of linear algebra is observed. As F_1 is pre-requisite for F_{13} andF_{13} is pre-requisite for F_{19}, which shows that F_1 is also pre-requisite for F_{19} *i.e.*, $F_1 \check{D} F_{19}$.Over all, we observe that for F_5 the security requirements S_4, S_5, S_6, S_{11} and S_{12} are essential and integral part for the security of function F_5.

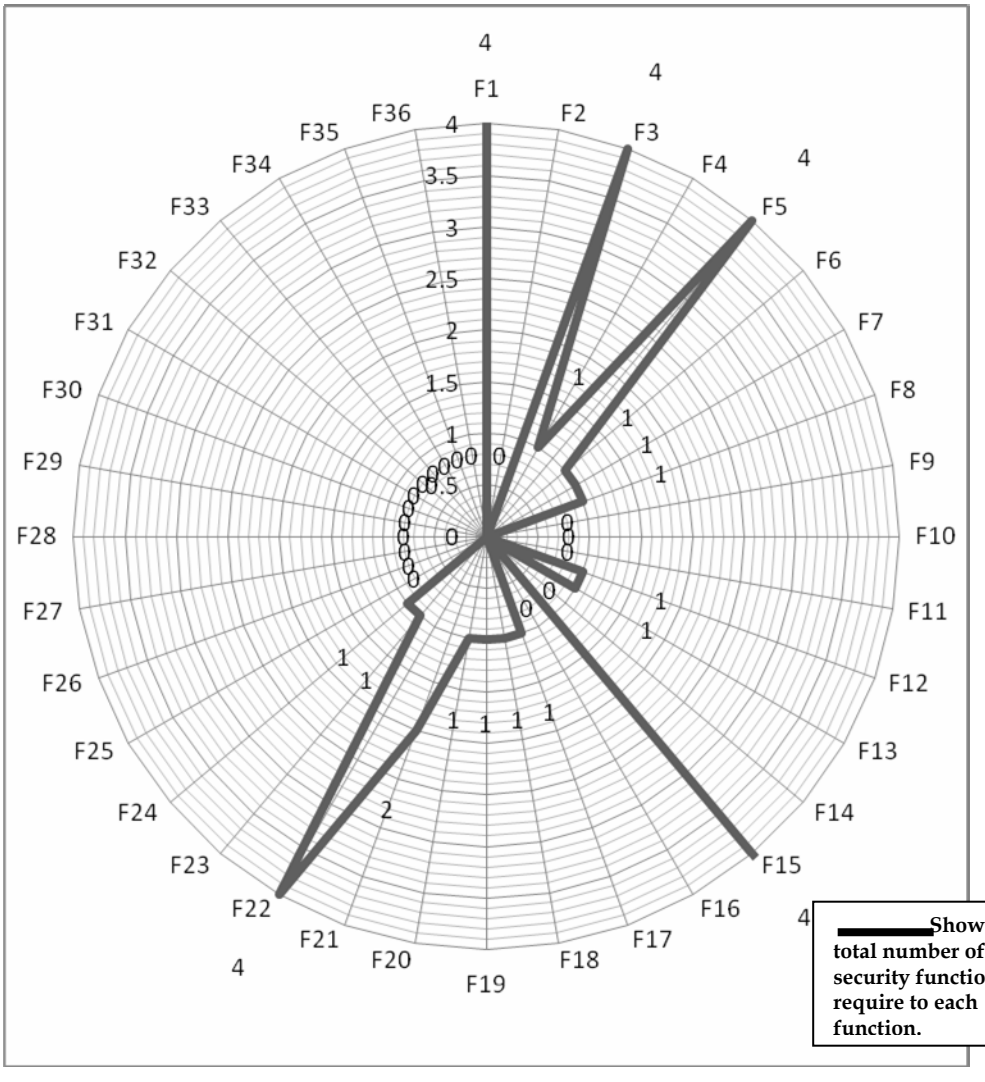

Figure 6: Security Feature N_d by Functional Requirements

Figure 6, shows the degree of security of different functions in terms of number of security features required for each of them. This graph infers the functions of the system, which are more security conscious. In future, we can also measure in terms of weight assignment, after assigning weight to each security feature. As we have noted that, there are a large number of functions for which security is a core element i.e. F_1, F_3, F_5, F_{15}, and F_{22} N_d is 4 security feature to make them secure. In Figure 6,the dark lines shows the graph of required security features whereas, the natural numeric shows total number of security function r. Definitely, if we follow this mechanism during SDLC, the security of the application will

improve. For comprehensive study and analysis of the security requirements of the E-legal system, see details at Appendix-1.

It has been observed from Table 3, that as we move towards latter stages of the application on such requirements, the security features decreases. However, the dependency increases to maximum at the end of system. Since the system already become secured, because of dependency among other secured functional requirement. It means requirement dependency is directly proportional to security requirements i.e., $\check{D}$ α Security.

As the requirements squeezes the dependency and security squeezes too. This shows that the security requirements are compulsory for different functional requirements. That means functional requirements and security requirements are inter-dependable.

Now, we will introduce mathematical model to incorporate security requirements of E-Legal system (given in Table 3) during development life cycle by using Requirement Analysis Algebra as a tool.

Table 4: Security Requirements of E-Filling

Dependence	**Security Requirement of E-Filing**		**Function**
	S_1	Submission required details for creation of account	F_1
S_1	S_2	Verification through e-mail (level-1):Verification through mobile call(level-2)	F_1
S_2	S_3	Verification through any other service such as SMS	F_1
	S_4	User identity	F_4
S_1	S_5	Grant of password	F_4
S_4	S_6	Change the system generated password	F_5
S_6	S_7	Alert generation for password change	
	S_8	On-line verification of User (CNIC)	F_6, F_7, F_8, F_{12}, F_{17}, F_{18}, F_{20}, F_{21}
	S_9	On-line verification of advocate from Bar Associations	F_1, F_{13}, F_{19}, F_{21}

Complete security requirements of the E-Legal system is given in Appendix-II, where as its Algebra is provided in Figure 8.

$$\ddot{v}(S_1)=F_1$$

$$\ddot{v}(S_2)=F_1$$

$$\ddot{v}(S_3)=F_1$$

$$\ddot{v}(S_4)=F_4$$

$$\ddot{v}(S_5)=F_4$$

$$\ddot{v}(S_6)=F_5$$

$$\ddot{v}(S_8)=\{F_6, F_7, F_8, F_{12}, F_{17}, F_{18}, F_{20}, F_{21}\}$$

$$\ddot{v}(S_9)=\{F_1, F_{13}, F_{19}, F_{21}\}$$

$$\ddot{v}(S_{11})=\{F_2, F_5, F_{15}\}$$

$$\ddot{v}(S_{12})=\{F_5, F_{15}\}$$

$$\ddot{v}(S_{13})=\{ F_3, F_{15}, F_{22}, F_{23}, F_{24}\}$$

$$\ddot{v}(S_{14})= \{F_3,F_{22}\}$$

$$\ddot{v}(S_{15})= \{F_3,F_{22}\}$$

$$\ddot{v}(S_{16})= \{F_3,F_{22}\}$$

Figure 7: Utilization of Security for Various Functions

Figure 7, shows that $S_8$is required by $F_6, F_7, F_8, F_{12}, F_{17}, F_{18}, F_{20}$ *and* F_{21}.Although these security requirements are also dependent upon each other, however we have not included this aspect presently in the *RAA* for the time being. The importance of user's functional and security concerns has forced us to use mathematical modeling techniques which provide the necessary foundation for software design. Moreover, by applying these algebraic functions, it helps us to identify the required security features.

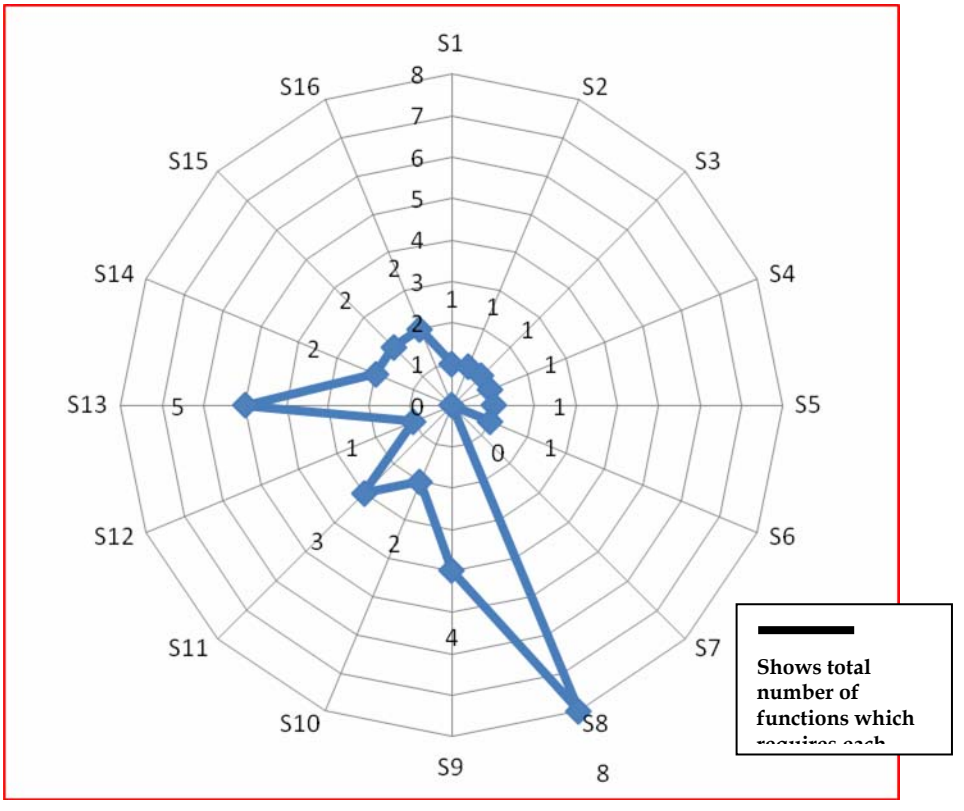

Figure 8: Graph showing Security Features needed to Various Functions

The security and functionality may be significantly improved using RAA while designing and developing a system demanding extensive security measures for its implementation specifically in Cloud Computing environment.

6) CONCLUSION AND FUTURE WORK

The proposed Requirements Analysis Algebra (*RAA*) presented in our research work addresses findings partially as further research is in progress. The research work is part of research carried out under title "A Secured Methodology for Designing Cloud Computing Applications" (Nasir A, 2013). The RAA proposed may be helpful in analyzing applications in the analysis phase. The salient feature of the algebra is that it built a table 1, of user versus their opted requirements using the *Opt* operator (see Section 4.3.4). The novelty of these research findings is that these can be used during the development phase and even later on when the application is operational. The use of the table 1, during operational

phase may have its implication in monitoring and tuning up the software applications in more rigorous way.

REFERENCES

Andreas Berl, Erol Gelenbe , Marco di Girolamo , Giovanni Giuliani, Hermann de Meer , Minh Quan Dang and Kostas Pentikousis (2010), "Energy-Efficient Cloud Computing" The Computer Journal, Vol. 53, pp. 1045-1051.

Anis Charfi and Mira Mezini (2005), "Using aspects for security engineering of web service compositions" in IEEE International Conference on Web Services, pp. 59–66.

Awais Rashid, P Sawyer, A Moreira, J Araujo (2003), "Modularisation and composition of aspectual requirements" in 2nd International Conference on Aspect-Oriented Software Development (AOSD'03) pp. 11-20.

Borko Furht, Armando Escalante (2010), Handbook of Cloud Computing, Springer Science, Springer street, New York, USA Pp. 24.

Dan Svantesson, Roger Clarke (2010), "Privacy and consumer risks in Cloud Computing", Computer Law & Security Review 26, pp. 391-397.

Feng Chen, Marcelo D. Amorim, and Grigore Rosu (2004),"A formal monitoring-based framework for software development and analysis, in Formal Methods and Software Engineering", 6th International Conference on Formal Engineering Methods 2004 pp. 357-372.

Hao Chen, David Wagner (2002), **"**MOPS: an infrastructure for examining security properties of software"**,** 9th ACM conference on Computer and communications security ACM 2002 pp. 235 – 244.

Hideyasu Sasaki (2011), "A computing theory for collaborative and transparent decision making under time constraint", Information Systems Frontiers, Vol. 13 pp. 207-220.

Holtzblatt K., Hugh K. Beyer(1995), "Requirements Gathering: The Human Factor",Communication of the ACM 38, pp.31-32.

Javier E., David C., Arturo M. (2008), "Application Development over Software-as-a-Service Platforms", in 3rd International Conference on Software Engineering Advances (ICESA 2008) IEEE Computer Society, pp. 97-104.

Jay Ligatti, Lujo Bauer, and David Walker (2005), "Enforcing non-safety security policies with program monitors In Computer Security" European Symposium on Research in Computer Security (ESORICS 2005), Vol. 3679 pp. 55-373.

Jianwei Zheng (2010), "The Evolution Process and Economic Analysis of Cloud Computing with Its Application in Chinese University", International Conference on Challenges in Environmental Science and Computer Engineering IEEE 2010, Vol. 2, pp. 361-364.

John Harauz, Lori M. Kayfman, Bruce Porrer (2009), "Data Security in the World of Cloud Computing" IEEE journal on Cloud Computing Security, Vol. 7, pp. 61-64.

Leavitt, N. (2009), "Is Cloud Computing Really Ready for Prime Time?", Journal of Computer (IEEE Computer Society), vol. 42, pp. 15-20.

Lori M Kaufman (2009), "Data security in the word of Cloud Computing" journal of IEEE Security & Privacy vol. 7, pp. 61-64.

Manal M. Yunis.A (2009), "cloud-free' security model for cloud computing", in International Journal of *Services and Standards*, vol. 5, pp. 354-375.

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, "A View of Cloud Computing", Communications of the ACM, 53: 50-58 (2010).

Mostert, D. (1995), "A Technique to Include Computer Security, Safety, and Resilience Requirements as Part of the Requirements Specification", Journal of Systems and Software, vol. 31 pp. 45-53.

Mythry Vuyyuru, Pulipati Annapurna, K. Ganapathi Babu, A.S.K Ratnam (2012), "An Overview of Cloud Computing Technology", International Journal of Soft Computing and Engineering (IJSCE), vol. 2 pp. 244-246.

Nasir, A (2013), Ph. D. dissertation "A Secured Design Methodology For Developing Cloud Applications", Department of Computer Science & Engineering, University of Engineering and Technology, Lahore, Pakistan (in progress).

NIST, (2009), "Cloud Computing Definition by National Institute of Standards and Technology", Publisher: NIST, pp. 53: 50.

Paolo Bresciani, Paolo Giorgini, Haralambos Mouratidis, and Gordon Manson (2004), "Multi-Agent Systems and Security Requirements Analysis", Software Engineering for Multi-Agent Systems II. Lecture Notes in Computer Science2940, pp. 35-48.

Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone (2005), "St-tool: A case tool for security requirements engineering", 13th International Conference on Requirements Engineering, IEEE pp. 451-452.

Paul T. Jaeger, Jimmy Lin & Justin M. Grimes (2008), "Cloud Computing and Information Policy? Center for Information Policy and E-Government" Journal of Information Technology & Politics, vol. 5 pp. 269-283.

Pethuru Raj, Mohanavadivu Periasamy (2011), "The Convergence of Enterprise Architecture (EA) and Cloud Computing", in the book Cloud Computing for Enterprise Architectures. Computer Communications and Networks published by Springer: Pp 61-87.

Playle G., C Schroeder (1996), "Software Requirements Elicitation: Problems, Tools, and Techniques", in the Journal of Defense Software Engineering, vol. 9 pp.19-24.

Qi Zhang, Lu Cheng, Raouf Boutaba (2010), "Cloud computing: state-of-the-art and research challenges", in the Journal of Internet Service and Application (JISA), vol. 1pp. 7-18.

R. Glott, Elmar Husmann, Ahmad-Reza, Sadeghi Matthias Schunter(2011), "Trustworthy Clouds Underpinning the Future Internet", The Future Internet, Lecture Notes in Computer Science, 6656, pp. 209-221.

Rachael King (2008), "Computing Heads for the Clouds", Business Week Magazine dated 08.08.2008.

Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic(2009), "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, published by Elsevier, 25, pp. 599-616.

Ran Li, Shoulian Tang, CeGuo, Xiaowei Hu (2010), "Thinking the cloud computing in China", Information Management and Engineering (ICIME), 2nd IEEE International Conference pp. 669-672.

Ricky W. Butler, James L. Caldwell, Victor A. Carreno, C. Michael Holloway, Paul S. Miner, and Ben L. Di Vito(1995), "NASA Langley's Research and Technology Transfer Program in Formal Methods" 10th Annual Conference on Computer Assurance (COMPASS 95).

Vinu George and Rayford Vaughn (2003), "Application of Lightweight Formal Methods in Requirement Engineering",Crosstalk in the Journal of Defense Software Engineering vol. 30.

APPENDIX-I

Depend	Functional Requirement of E-Filing		Security
Input	F_1	Provide user's information	S_1, S_2, S_3, S_9
	F_2	Selection of Electronic Facilities (E.F.)	
F_2	F_3	Payment of E.F. Charges	$S_{13}, S_{14}, S_{15}, S_{16}$
F_1	F_4	Generate User-id	S_{11}
F_4	F_5	Access the system through user's account _login	S_4, S_5, S_6, S_{12}
F_2	F_6	Access the information through CNIC from National Database.	S_8
F_2	F_7	To intake Applicant info on the basis of CNIC from Database	S_8
F_7	F_8	Submission of respondents' information using CNICs	S_8
	F_9	Provision of relevant law	
	F_{10}	Provision of relevant citation	
	F_{11}	Verification of relevant law/citation	
F_4	F_{12}	Submission of Advocate Identity	S_8
F_{12}	F_{13}	Verification of advocate identity	S_9
F_4, F_7	F_{14}	Attach relevant documents & citations	
F_7, F_{14}	F_{15}	Submission of evidences, for example in criminal case FIR, report of medical and forensic science laboratory, DNA test etc. and in case of civil/family property, registration, Fardmalkiat, transfer deeds and marriage certificate, departments etc.	$S_{10}, S_{11}, S_{12}, S_{13}$
	F_{16}	verification from relevant departments	
F_7, F_8	F_{17}	Submission of Undertakings	S_8
F_7, F_{13}	F_{18}	Submission of Oath	S_8
F_{13}	F_{19}	Signatures of advocate	S_9
F_7	F_{20}	Signature of litigant	S_8
$F_{7,13}$	F_{21}	Verification of Signatures of advocate and litigant public.	S_8, S_9

Depend	Functional Requirement of E-Filing		Security
F_2, F_{11}	F_{22}	Payment of court fee	S_{13}, S_{14}, S_{15}, S_{16}
F_{22}	F_{23}	Issuance proof of court fee	S_{13}
F_{23}	F_{24}	Verification of court fee	S_{13}
F_{15}	F_{25}	Issuance of unique diary	
F_{11}	F_{26}	Assign relevant law	
F_{25}	F_{27}	Checking of all requisite documents	
F_{26}	F_{28}	Checking of relevant provisions of law/citations	
F_{15}	F_{29}	Point out deficiency, if any	
F_7, F_{12}	F_{30}	Intimate/notify to the concerned	
F_7, F_{12}	F_{31}	Receive objections	
F_{27}	F_{32}	Address/remove the objections	
F_5, F_6	F_{33}	Re-submission after removal of objections	
F_{28}	F_{34}	Issue the unique case No.	
F_{34}	F_{35}	Marking of the case to the relevant bench/judge as per category	
F_{35}	F_{36}	Fix the hearing date	

APPENDIX-II

Dependence	Security Requirement of E-Filing		Function
	S_1	Submission required details for creation of account	F_1
S_1	S_2	Verification through e-mail (level-1):Verification through mobile call(level-2)	F_1
S_2	S_3	Verification through any other service such as SMS	F_1
	S_4	User identity	F_4
S_1	S_5	Grant of password	F_4
S_4	S_6	Change the system generated password	F_5
S_6	S_7	Alert generation for password change	
	S_8	On-line verification of User (CNIC)	F_6, F_7, F_8,F_{12}, F_{17}, F_{18}, F_{20}, F_{21}
	S_9	On-line verification of advocate from Bar Associations	F_1, F_{13}, F_{19}, F_{21}
S_4	S_{10}	Access to the system	F_2, F_{15}
S_4	S_{11}	Submission of user through ID	F_2, F_5, F_{15}
S_4	S_{12}	Verification of user through password (level-1)	F_5, F_{15}
		Verification of user through digital Figure (e.g. Captcha) selected characters of the password (Even next Level can be added to restrict cut, paste etc.)	
S_{10}	S_{13}	Checking of availability of funds in the relevant account	F_3, F_{15}, F_{22}, F_{23}, F_{24}
S_{11}, S_{12}	S_{14}	Transfer of funds through password	F_3, F_{22}
S_{11}, S_{12}	S_{15}	Use of some smart/credit card etc.	F_3, F_{22}
S_{14}, S_{15}	S_{16}	Payment of court fee through credit card	F_3, F_{22}

APPENDIX-III

Functional Requirement of E-Filing		Input	Output	Consequences	Security
		Data/Information Type	Data/Information Type		
F_1	Provide user's information	Secret	Secrete	Confidentiality	High
F_2	Selection of Electronic Facilities (E.F.)	Open	Open	Integrity	Medium
F_3	Payment of E.F. Charges	Open	Secrete	Integrity	High
F_4	Generate User-id	Top Secret	Top Secret	Confidentiality	High
F_5	Access the system through user's account _login	Secret	Secrete	Confidentiality	High
F_6	Access the information through CNIC from National Database.	Open	Open	Availability	Low
F_7	To intake Applicant info on the basis of CNIC from Database	Open	Open	Availability	Low
F_8	Submission of respondents information using CNICs	Open	Open	Availability	Low
F_9	Provision of relevant law	Open	Open	Availability	Low
F_{10}	Provision of relevant citation	Open	Open	Availability	Low
F_{11}	Verification of relevant law/citation	Open	Open	Availability	Low
F_{12}	Submission of Advocate Identity	Close	Open	Integrity, Availability	Low
F_{13}	Verification of advocate identity	Close	Open	Availability	Low
F_{14}	Attach relevant documents & citations	Open	Open	Availability	Low
F_{15}	Submission of evidences, for example in criminal case FIR, report of medical and forensic science laboratory, DNA test etc. and in case of civil/family property, registration, Fardmalkiat, transfer deeds and marriage	Close	Close	Availability	Low

Functional Requirement of E-Filing		Input	Output	Consequences	Security
		Data/Information Type	Data/Information Type		
	certificate, departments etc.				
F_{16}	verification from relevant departments	Close	Close	Integrity, Availability	High
F_{17}	Submission of Undertakings	Close	Close	Integrity, Availability	High
F_{18}	Submission of Oath	Close	Close	Availability	High
F_{19}	Signatures of advocate	Open	Open	Availability	High
F_{20}	Signature of litigant	Open	Open	Availability	High
F_{21}	Verification of Signatures of advocate and litigant public.	Close	Close	Integrity, Availability	Low
F_{22}	Payment of court fee	Open	Open	Availability	High
F_{23}	Issuance proof of court fee	Close	Close	Integrity, Availability	High
F_{24}	Verification of court fee	Close	Close	Integrity, Availability	High
F_{25}	Issuance of unique diary	Close	Close	Integrity	High
F_{26}	Assign relevant law	Open	Open	Availability	Low
F_{27}	Checking of all requisite documents	Secret	Secrete	Availability	High
F_{28}	Checking of relevant provisions of law/citations	Open	Open	Availability	Low
F_{29}	Point out deficiency, if any	Open	Open	Availability	High
F_{30}	Intimate/notify to the concerned	Open	Open	Availability	High
F_{31}	Receive objections	Open	Open	Availability	Low
F_{32}	Address/remove the objections	Open	Open	Availability	High
F_{33}	Re-submission after removal of objections	Close	Close	Availability	High
F_{34}	Issue the unique case No.	Open	Open	Availability	High
F_{35}	Marking of the case to the relevant bench/judge as per category	Open	Open	Availability	High
F_{36}	Fix the hearing date	Open	Open	Availability	High