Web Service Security Through Business Logic

M.S.SALEEM BASHA *
Department of Computer Science
Pondicherry University
R.V Nagar, Kalapet, puducherry, india

M.THIRUMARAN
Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry, India

P.DHAVACHELVAN
Department of Computer Science
Pondicherry University
R.V Nagar, Kalapet, puducherry, india

***Abstract*—Service Computing has recently gained significant momentum to enable the IT services and computing technology to perform business services more efficiently and effectively. Presently the service computing concentrates on inventing and establishing business models which leads to better QoS driven targeting services. So replacing existing business model by QoS driven business model would be the future preference of service computing environment. Further, the most critical and challenging part of web service computing is handling of non functional aspect that may occur during service composition and service execution process. There are many non-functional aspects are involved during service composition and execution at run time. One such a business model solution is called Business Logic Model (BLM) which is proposed in this paper, the primarily aim is to design and implement the business logic for functional and non functional aspect targeting successful service computation. One of the important feature of this model is to manage exceptions occurs during service composition in execution time. The distinct ability of the proposed model is managing the runtime exception is proved with the help of a set of QoS parameters that are especially satisfying the service computational criteria such as computability, traceability, security and decidability.**

Keywords-Business logic model; Web service Security; Quality of service; Computational criteria.

I. INTRODUCTION

Organization faces the problem of the security derived from the non functional requirements and to maximize the utilization of the cutting edge technology with minimum cost in the agile business environment. Web service is the upcoming wave for tomorrows business needs, in this concern the non functional attributes is the one of the major challenging sector for the developers to guarantee the confidentiality, authentication, integrity, authorization and non-repudiation of machine to machine interaction so security is not negotiable to anticipate a secure artifacts for web service. There are two underlying themes for all these pressure: Heterogeneity and agility: Software development is a standard practice in software engineering where business logic drives the software development starting from requirement analysis to maintenance. The information exchange between the database and the user interface will be done by the functional algorithm which is described by the business logic. This logic is composed of business functions and business rules. Series of logically related activities or task performed together to produce a defined set of result called business function and business rule is a statement that defines or constrains some aspect of the business. It is important to understand that business modeling commonly refers to business process design at the operational level [4] which comes under the functional requirement of the system, where as the non functional requirements are left as it is afterthought. Non functional attributes are defines system properties and constraints and can be classified as Product requirements, Organizational requirements and External requirements. Security of the system plays a major role across the boundaries of the organizations. Security of the system can be improved by providing the foundation in the early phase of the system development process. The development of system during requirements analysis and system design can improve the quality of the resulting system.

II. Business Model

A business model describes a company’s operations, including all of its components, functions and processes, which results in costs for itself and value for the customer. Therefore, it is how the engine for the business actually works. The objective is to have low cost and high value and thus maximize profit. All the business models are not equal in respect to the business. All the business models has some unique secret and the investor’s task to find it out. More importantly, it is to discover the pixie dust that others cannot see [25]. The uniqueness of business model will be determined by combining the four aspects of productions, namely land, labor, capital and enterprise. The competitive advantage of the sector will determine the ability of the organization to achieve the business model.

Business modeling of a system improves the quality and cost effective of the system. Business are learning that security in their electronic processing and electronic storage as well as security in their interaction among customers, business partners, suppliers and service organizations is the essential ingredient underpinning electronic business models. The need for trust will only increase as new logarithmically growing, paradigm-shifting technologies proliferate, mutating business models and IT operations and management models further, while introducing new risk and vulnerabilities. The confidence in the security models is needed for all the players in the business model which drives the as a force for the prominent business for economic prosperity.

III. Business Logic Model

Enterprise systems are distinct and highly complex class of systems. They are characterized by their importance for enterprises themselves, making them mission critical, by their extreme multi-user capability, by their tolerance of heavy loads and by with their tight integration with the business process, which makes every enterprise system installation unique. In short, they are one of the most fascinating yet most demanding discipline in software engineering [1]. The main consumer of the service that the data layer provides is the business logic. The business logic is responsible for implementing the basic rules of the system according the operating rules of the business. Its main feature is to take request, determine what actions the request requires, implement those actions and return response data to the customer.

Modeling business logic focuses on the core functionality of the business process, which are capsulated as web services. It requires that business process pertains exactly to the business logic with various business terminologies such as policy, standards, constraints, etc. As a prerequisite to this business logic model, the core functionality of the business process should be analyzed then modeled absolutely, whereas the previous implementations of web services were direct. Ronald et al. states that existing models like business rule model, business motivation model and business process model concentrate on business process at the operational level with compromising minimum range of QoS attributes [2]. Business rule model deals with the extraction of business rules from the business logic, in order to reduce the cost and time spent in development [2][3]. Business motivation model paves way for identifying the facts preserved in novel objectives, thereby facilitating the business process development. Business process model provides optimization to the business process at the designing phase. The implementation of a company's business model into organizational structures and systems is part of a company's business operations. It is important to understand that business modeling commonly refers to business process design at the operational level [4], whereas business models and business model design refer to defining the business logic of a company at the strategic level.

Business logic model aims to resolve the complexities involved, by decomposing the business process into sub processes and in turn into tasks, also preserving the functional dependencies among the sub-processes, without ignoring the key factors. Any service domain adopted this model for their web service development could be easily managed in terms of handling run time exceptions towards service reliability and manageability. Business logic model can be applied in tandem with the above described models, thereby facilitating service computation and composition much better. This model enables web services to realize their computational criteria such as computability, traceability and decidability with the supporting QoS attributes like manageability, configurability, serviceability and dependency. The computational criteria would be the best suit for the web service community who look for exception-free web services or reconfigurable web services. This model would also satisfy the service consumers who approach the discovery and composition engines for fetching exception free or self configurable web services. Hence this model would ensure the consumers that the services are manageable at runtime, self configurable in case of exception, computable in total or partial and traceable to the point of failure. Also it sustains dependency between the business rules and business functions.

Modeling system with business logic model has benefits like; it reflects standard layering practices with in the development communities, business functionality easily accessible by other object application, very efficient to build business objects, it helps to test the basic success premises of business, improves the clear understanding of existing value drivers and constraints, it provides a componentized view of the business and technology environment in order to have common building blocks that can be reused across product and business silos, it defines and sustainable interim states which provides measurable benefits as flexible path to the goal and business logic provides a strong governance to manage and deliver the changes. Business logic also possesses some of the drawbacks; significant performance problem for data intensive functions, non object application may have significant difficulty to accessing functionality. Improper handling of the non functional requirements may result in compromising the growth of the organization.

Currently much work in the requirements engineering field has been done to shown the necessity of business logic which take non-functional requirements (NFR) into consideration. Such logic will better deal with real-world situations. On the other hand the advantages of having business logic is the capability of representing nonfunctional aspects, such as confidentiality, performance, ease of use and timeliness. It is believed that these functional aspects should be dealt with as non-functional requirements. Therefore, NFRs have to be handled and expressed very early in the process of modeling an information system [5]. Organizations are spending much in system development and least concentration to NFRs. Recent tales of failure in information systems can be explained by the lack of attention to NFRs. The London Ambulance System (LAS) is an example for the information system failure due to lack of attention of NFRs [6]. The LAS was deactivated, soon after its deployment, because of several problems, many of which were related to NFRs such as performance and conformance with standards [7]. Negotiation in the NRFs is not a healthy activity in the system development, the consequences of negotiating NRFs leads to serious problem as in the case of LAS. Serviced Oriented Architecture(SOA) is the paradigm for the future business environment, where web service is the building block for SOA and it is the key for agile business across the enterprises. It is important in Service Oriented Architecture to separate functional and non-functional requirements for services because different applications use services in different non-functional contexts. In order to maximize the reusability of services, a set of constraints among non-functional requirements tend to be complicated to maintain. Currently, those non-functional constraints are informally specified in natural languages, and developers need to ensure that their applications satisfy the constraints in manual and ad-hoc manners [8].

System developers believe that business logic composes and speaks only the functional aspect, but fails to keep in mind that to consider the other aspects driven by functional aspect i.e. non-functional aspect. The separation of functional and non-functional aspects improves the reusability of services and connections. It also improves the ease of understanding application design and enables two different aspects to evolve independently. Wada et al. pointed that the separation of functional and non-functional aspects, results in higher maintainability of applications [9]. Non-functional aspects should also be captured as abstract models in an early development phase and automatically transformed to code or configuration files in order to improve development productivity. It incurs time-consuming and error-prone manual efforts to implement and deploy non-functional aspects in later development phases (e.g., integration and test phases) [10][11]. Web services become more popular and better utilized by many users and software agents, they will inevitably be commercialized. But still Services Challenge (WSC) that focus on functional aspects [12][13]. We believe that considering both functional and non-functional attributes together in solving the Web services composition problem would produce superior outputs [14]. Because NFRs are always tied up with functional requirements i.e., NFRs can be seen as requirements that constrain or set some quality attributes upon a functional requirements [26].

To the best of our knowledge, this is the first work to study the usability of the main approaches adopted for specifying and enforcing web service security requirements in business logic. Today’s internet and e-affaires are the composite blend of business process and technology where the web service is the perfect blue print for agile business environment. In the early times, data in the networks were closed; security within these networks was ensured through isolation. Later LAN(Local Area Network) was introduced with firewalls to isolated from the untrusted public networks to ensure that adversaries and hackers cannot intrude into the private network. For more security they added security aspects like proxies, intrusion dectection system, intrusion prevention system, antivirus, malware catchers etc., are the domain specific security measures. The belief was that applications and assets used by the organization can be secured through in-vitro perimeter security. Therefore, software engineering techniques never looked into security as an important component in Software Develop Life Cycle

(SDLC); and, identified security as nonfunctional requirement [15]. Security must be part of the application to protect itself from security threats. Application security will however be over and above the perimeter network security. To achieve this, security now need to be treated as functional requirement and must be part of SDLC [16]. Sindre et al [23]. have identified application security as a need and proposed ways to achieve this. All these isolated and independent techniques have been combined together in a thread to form a business Logic [17].

A. Relation between models

Business logic is composed of some ingredients such a business process, business functions, and business rules. A business process is a set of activities or functions defined by the experts which reflects the workflow of the organization towards a specific goal, an improper modeling of business logic may lead to increase the operational cost and hence it affect multiple application beneath this. Business process speaks the current satiation of the organization to analyze and enhance. Business Process Management (BPM) is a methodology that initiates the enterprises to specify business process incrementally. BPM also addresses how organizations can identify, model, develop, deploy and manage their business process[27]. Business function defines “what a system is supposed to do”, all the activities of the business process is reflected in the business function. Many business functions are combined to form a business process. Business function is a functionality that handles the modules, those modules partially fulfills of the business goal. Business rule is the fine grained of the business process which takes the real time parameters and variables. The related phases and goals of the business models are shown in figure 1.

The above systems have been adopted in many industries, however the critical issue and key properties of business logic have not been considered in detail. The business logic layer in N tier architecture composes of various business process, business functions and business rules. Each component in the business layer is implemented usually as monolithic.

B. Significance of BLM

Business processes and motivation models have been used to analyze and propose new changes in accordance to changing business scenarios. A process model scope does not extend optimally to web services, whereas Business Rule models extract rules from the business logic and concentrate mainly on the problem of modeling and accessing data by using efficient queries [28][2]. However they do not model the entire business logic. Thus there is a need for a model which represents a business process in detail and also adapts the rules, policies and standards to changing business scenarios. This adaptability helps service consumers and service providers cope up with the demanding and challenging changes in services. Such a representation should not compromise on matter and processes private to a business. Since a business logic model seems inevitable, by maintaining business privacy and by modeling a specific business process, the model seems to be a promising methodology to handle the ever-changing business scenarios. Business Process systems that use web services decrease the cost of automating transactions with trading partners. The scope of a business process is limited to design, development and deployment of services. The limited scope helps to develop better services keeping service customization in mind.

The outcome breakdown structure of the service business logic is streamed as a set of business rules, functions and parameters. Further, these rules and functions could be tuned to be primitive business functions under certain specific conditions. The primary motivation behind setting up the business functions as primitive business functions would pose the computability and traceability factors, which are the most essential quality-driven factors as they could manage the complete service computing platform successfully by the effective handling of run-time exceptions during service computation and composition. This model decomposes the business logic into functionally consistent and coherent business rules and functions, keeping in mind the privacy constraints of Businesses. Decomposition helps representing the interdependent business functions. This strategy categorizes the business functions into initial, composite and recursive functions and evaluates them into computable and traceable business functions. Computability and Traceability of business functions are key factors for measuring customization. Existing discovery and composition engines provide services based on functionality, quality, and security of requested services. Customizing the services is not addressed by the existing engines. The proposed business logic model exhibits the functionalities of any of the generic engines but is also resilient to customization.

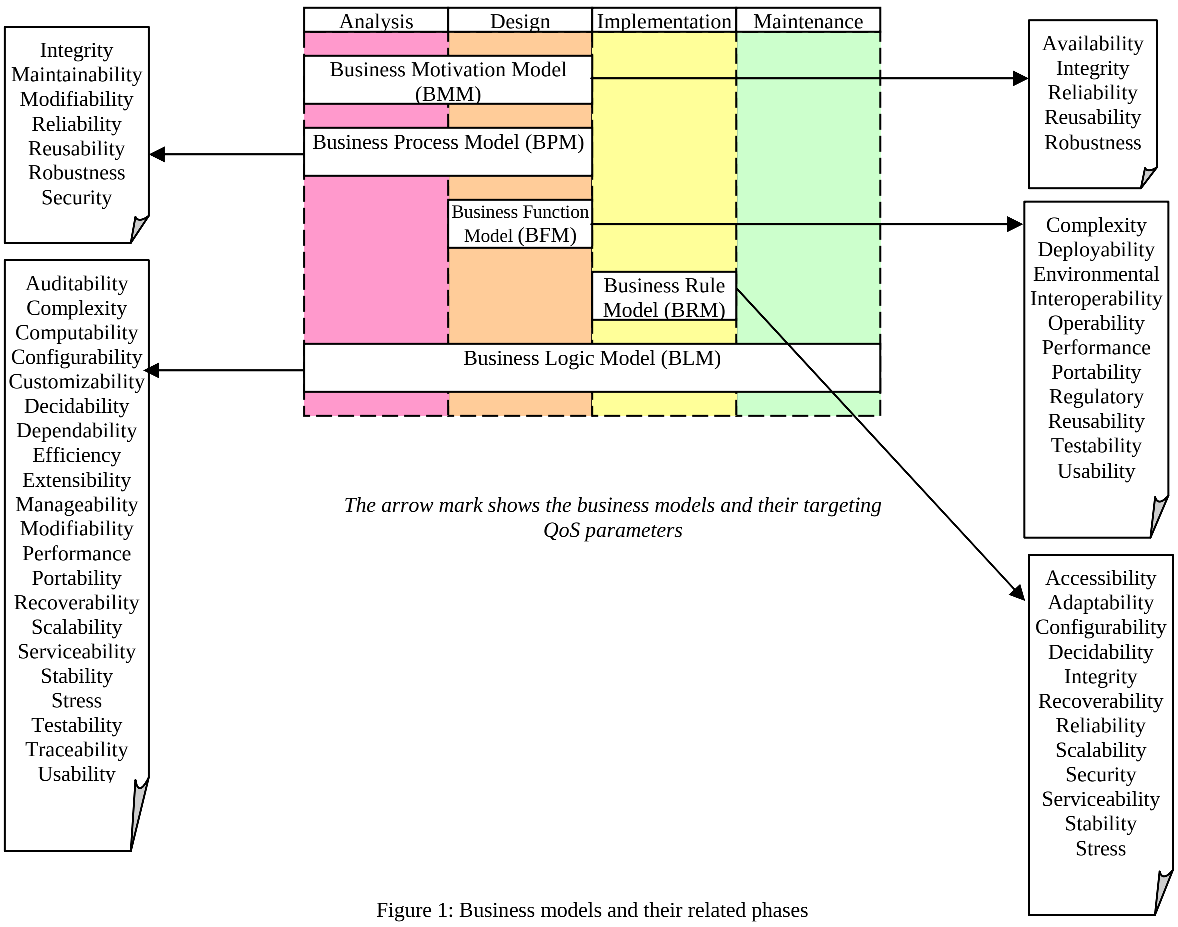


Figure 1: Business models and their related phases

C. Significant of business logic in security

In web service, the non functional attributes such as security, QoS, and fault tolerance of web service and the connection should be defined in the business logic. The separation of functional and non functional aspects improves the reusability of service and connection [19]. Of course the separation may improve the reusability and connection, but in the context of security non functio*Figu*nal aspect is the sub set of business logic and driven by functional aspect. Incorporating security as non functional aspect in business logic gives the broaden vision of the system development and gives no rooms for afterthought. Business logic is the composition of evaluations, transformations, decisions, transitions, request and response to complete a task of the organization. Fabien Baligand et al have worked towards the weaving of non functional security aspects depending on policies requirements to achieve a genuinely flawless web service. W3Consortium, UPnP, DPWS(Device Profile for Web Service) and OSGi are the various protocols and mechanism have been introduced in recent years for describing, publishing and composing web services. In accordance with the composition of web service need to achieve a set of non functional aspect related to security or availability[20]. Business logic imposes constrains of the analysis to maintenance phase. Many believe that business logic deals only with functional aspect, but as a whole business logic is the mirror image of real business, non functional aspect defines "how the system is supposed to do" hence negotiating the non functional aspect is not a healthy activity. For example to develop a secure banking system there may be many business functions (Deposit, Enquiry, Withdraw, etc.) and also security attributes (authentication, authorization, etc.,) are one of the major requirements for banking system which falls under the non functional requirements. Negotiating this non functional requirements may lead the enterprise to face the serious consequences and heavy revenue loss and the must pay heavy cost more than the loss to recovery. So it is suggested that non functional aspects should treat as functional aspect in business logic.

D. Business Logic Model Proposal and Formulation

The proposed business logic model has three major components such as Business Logic analyzer, Business Rule Engine and Business Function Adapter as illustrated in figure 2. The core business logic corresponds to the particular project domain and is analyzed and decomposed into a reduced set of functional and non functional aspect of business logic by the business logic analyzer and the decomposer components. Business Rule Engine will extract the set of rules from the business logic and submit them to the next component called Business Function Adapter (BFA). BFA is the core component of the proposed model, performs extended analyses on the business logic and the extracted business rules to identify the business functions associated with the logic. It has various sub components like planner, Business Function (BF) classifier, BPEL engine, Computational engine and Schema Generator. Planner makes a decomposition plan to reduce the complexity of business logics and business rules. Mapper makes the mapping action non functional aspect associated with functional aspect. BF Classifier uses Business Function Adapter (BFA) algorithm to identify and categorize the business functions in terms of initial, composite and primitive recursion. The BPEL engine controls the workflow of the business functions with respect to the business logic. The computational engine is meant for evaluating the quality parameters according to the computational criteria enforced by the BFA algorithm. Finally the schema generator generates the Business Logic Schema which provides the complete description of business logic in XML format. This schema gives the authority and understandability of how the logic is decomposed and implemented.

Steps for Business Logic Modeling

(i) Analyze business logic (BL), identify and extract functional and non functional business rules (BR) and its associated business functions (BF).

(ii) Categorise the initial, partial, and recursive business function w.r.t primitive recursive function (PRF).

(iii) Represent the complex and nested business functions in terms of Primitive Business Function (PBF).

(iv) Ensure the QoS requirements of computable business functions.

(v) Note: Properties of primitive business functions include computability, traceability, security and decidability.

The identification and extraction of business functions with respect to the functionality of the business logic is done by Business Function Adapter Algorithm. It classifies identified business functions into initial, composite and primitive recursion through the traditional the mathematical technique called “Primitive Recursive Function”. PRF is primarily used for deciding the particular arithmetic or whether the logical problem is computable or not. It can also be used for deciding whether the programming logic is computable or not since it has the default characteristics to verify the initial and boundary conditions required for any type of computation. In this paper we propose a new function called “Primitive Business Function (PBF)”, which is actually originated from the PRF for specifically evaluating the business logic and its various supporting resources. The target of PBF is to ensure and evaluate the computational criteria such as computability, traceability, security and decidability. In N-tier programming paradigm, PBF would be the powerful solution to ensure the above listed quality attributes. Errors and exceptions are the two big black holes of any software product. There are a number of mechanisms existing for managing these issues at the maintenance level but there is no standard way to defend it to some extent at the design level. The proposed model attempts to overcome the problem by establishing the business logic model from the analysis to maintenance phase efficiently and effectively.

E. Analyzing Business logic

An organization has a business logic which is paramount to the organization. Any change in the physical world affects the organization’s business logic. In simple terms the most important component of the organization is affected thereby forcing the management to run maintenance steps to cope with the changing times and situations. But by the business logic adaptor, the business logic is split up into numerous numbers of simple parts which can be maintained with ease. This business logic adaptor analyzes and categorizes various business functions of the business logic with respect to the current strategy (in real -time). Typically business logic consists of number of business functions, which are all computable at reasonable costs within the expected time to achieve the goals. Since the business functions need to take many guesses in various processing stages, it leads to partial computation in most of the cases, where it actually aimed of total computation. Predicting whether the identified business logic is partially computable or totally computable is a very important factor to be considered for the ever changing business context. Initial stage of this model is to identify business logic which is going to meet certain business parties’ need with the effect of various business factors which lie on the current business solution. Ultimately identifying business logic with respect to current business strategy is the most critical and common issue which perhaps decides the success or failure ratio more abstractly. Once the business logic is defined, the real scope of business logic modeling process begins and it is meant to analyze, classify and isolate business functions from the defined business logic.

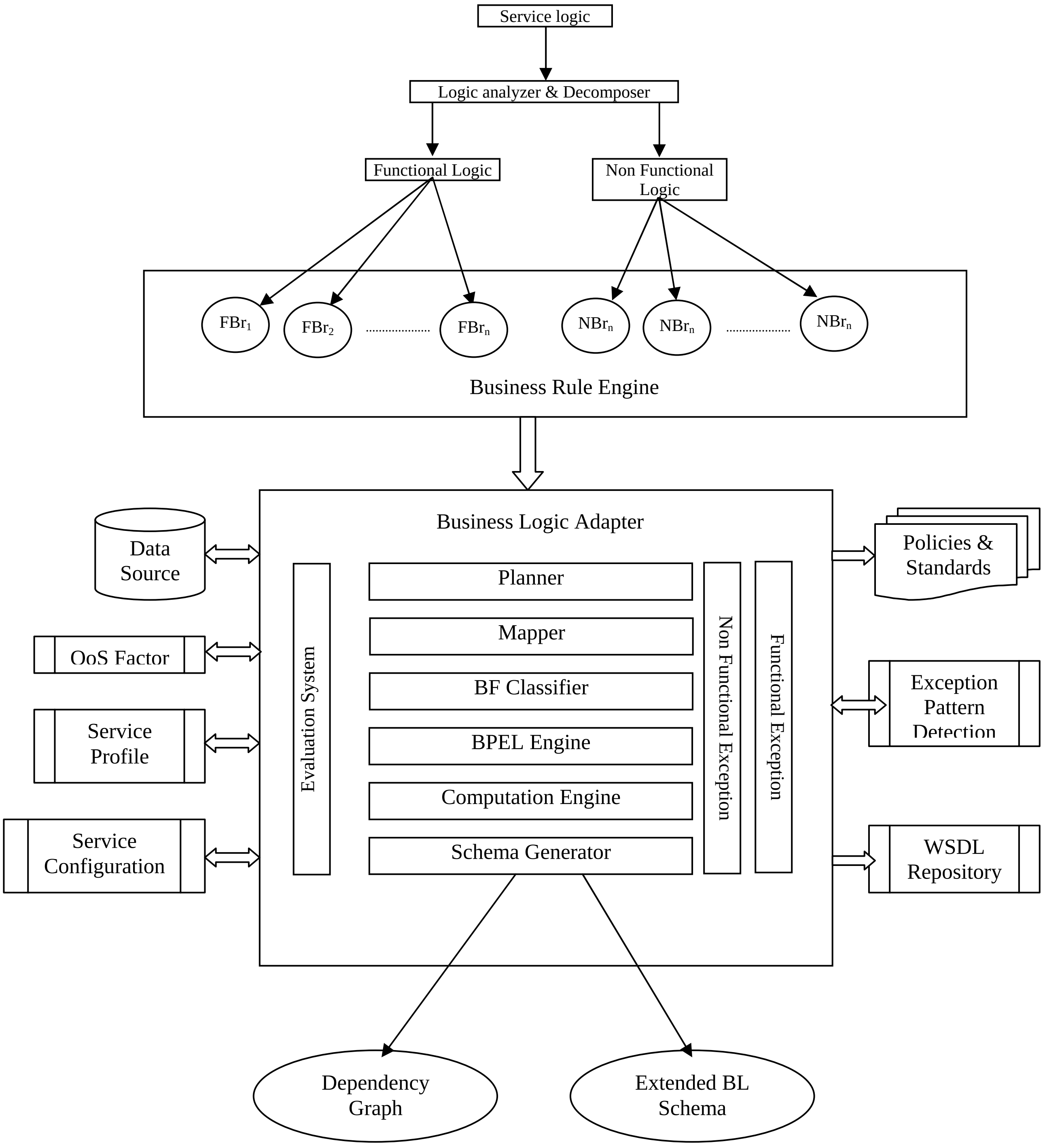


Fig 2: Proposed Business Logic Architecture

This business function identification is the key process applied to the business queries without violating the generative power of business logic. By decomposition Rule:

G(ßL)= G(ßf1,ßf2…. ßfn)
Where G(ßL) is the generative power of business logic and G(ßf1,ßf2…. ßfn) is the generative power of business functions.

After the identification and isolation of business functions are over, we step into the representation stage where business functions are fine-tuned and represented in terms of initial business functions, composite business functions and business functions by primitive recursion. This representation has always been the formal modeling technique traditionally followed by the primitive recursive function. Finally we propose the new term primitive business functions for business scenarios which are computable function since they are originated from primitive recursive functions. Hence in the next stage we can easily estimate the computational complexity of the defined business functions. It could be within the polynomial range of time. If the time bound is not satisfactory the business functions can be further refined with flexible business parameters and redefined to meet the expected time bound. This process is repeated until an effective business solution is obtained. Finally, the business solution is delivered in terms of Business Logic Schemas with its performance measure. This performance measure decides the computability of all of the business functions concerned by the whole business logic. Finally business parties can predict the success or failure ratio of business logic what they are enforcing on demand.

F. *Mathematical Methodology for Business Logic Modeling*

Deriving Primitive Business Functions from Primitive Recursive Function:

We identified the following initial business functions at the preliminary level. We can find and list out many other initial functions in this way. The below listed mathematical functions are used to return some values based on the input and context. These initial functions have been structured to map with various types of exception. For example, the Zero function Zßf (x) returns ‘0’ for any given value of ‘x’. Exceptions like ‘divide by zero’ can be resolved using zero function.

Zero function : Zßf (x)= 0
Unit function : Unßf (x) = 1
Identity function : Ißf(x) = x
Projection function : Prßfin(x1, x2, x3,… xn) = xi
Select function : Slßfi,j,kn(x1, x2, x3,… xn) = [xi ,xj ,xk]
Successor function : Sßf (x) = x+1
Predecessor function : Pßf (x) = x-1
Constant function : Cßf (x) = x’
Unique function : Ußf (x) = Ußf (y) iff x=y for all x,y
Equal function : Eßf (x,y)= Eßg (x,y) for all x,y
Insert function : Inßf (x1, . . . , xn,y)= (x1, . . . , xn,y)
for all values of x1, . . . , xn
Delete function : Dlßf (x1, . . . , xn, xi)= (x1, . . . , xn-1)
Update function :Upßf (x1, . . . , xn, xi)= (x1, . . xi . , xn)
Get function : Gßf (x) = y iff x= Gßf -1 (y)
Return function : x= Rßf -1 (y) iff Rßf (x) = y
Null function : Nßf (x) = • where • is null symbol Iterated exponentiation:
ßf (x,y) = xy
where xy = (….((xx)x)…..)x take x0 =1

Composition: Let ßg1, . . ., ßgm be primitive recursive functions, each of arity n, and h be a primitive recursive function of arity m. Then the function ßf with ßf (x1, . . . , xn) = ßh(ßg1(x1, . . . , xn), . . . , ßgm(x1, . . . , xn)) is also primitive recursive.

μ-Recursion: Let h be a (partial) function of arity n+1. A function f of arity n is defined by μ-recursion from ßh if ßf (x1, . . . , xn) = sμ y[ßh(x1, . . . , xn, y) = 0] , where sμ y[ßh(x1, . . . , xn, y) = 0] = z if ßh(x1, . . . , xn, z) = 0 and for all y0 < z, ßh(x1, . . . , xn, y0) is defined and ßh(x1, . . . , xn, y0) , 0. If no such z exists, sμ y[h(x1, . . . , xn, y) = 0] is undefined. ßf (x,y)= ßh (ßg1 (x,y), ßg2 (x,y)) from defined functions ßg1, ßg2, ßh.

Business function by Primitive Recursion, by which a function can be defined recursively through

ßf (x,0) = ßg1 (x)

ßf (x,y +1)= ßh (ßg2 (x,y), ßf(x,y)) from defined functions ßg1, ßg2, ßh Finiteness: f(x1, . . . , xn) =1 if n be natural number 0 otherwise (infinite)

Composite Business Function, by which we construct

Emptiness : f(x)= 1 if x = Ø

Equivalent function: Eqßf (x,x) = 1 for all x (Reflexive) | Eqßf (x,y) = Eqßf (y,x) (Symmetric) | Eqßf (x,y) + Eqßf (y,z) = Eqßf (x,z) (Transitive)

```
**Algorithm Business_Function_Adapter**(Business logic , Business Object)
// Input : Business logic [BL], Predefined Business Objects[BO] and parameters[BP].
// output: Functionally computable; non functionally computable
// Ißf[i] : Array for Initial Business Function, where i >= 1.
// SFßf[i] : Array of Simple Functional aspect of Business Function, where i >= 1.
// CFßf[i] : Array of Composite Functional Aspect of Business Function, where i >= 1.
// RFßf[i] : Array of Recursive Functional aspect of Business Function , where i >= 1.
// SNFßf[i] : Array of Simple Non Functional aspect of Business Function, where i >= 1.
// CNFßf[i] : Array of Composite non functional aspect of Business Function , where i >= 1.
// RNFßf [i] : Array of Recursive Non Functional aspect of Business Function, where i>=1.
// Fßr[i] : Array of Functional Business Rules, where i>=1.
// NFßr[i] : Array of Non Functional Business Rules, where i>=1.

Begin:

(i) Input_BusinessLogic(BL)
(ii) Identify and Extract Business Parameters
(iii) Divide BL ->   $FBL_1,FBL_2,…FBL_n$ such that all identified BL's [$FBL_1,FBL_2,…FBL_n$ ] functionally depends
on BL.
(iv) Divide BL-> $NFBL_1,NFBL_2…………NFBL_n$ such that  all identifiers are non functionality depends on BL.
(v) Extract $FBL_1….FBL_n$ -> $FBR_1,FBR_2……FBR_n$ such that all identifier [$FBR_1…..FBR_n$] are the functional rules
of BL.
(vi) Extract $NFBL_1….NFBL_n$->$NFBR_1,NFBR_2……….NFBR_n$ such that all identifiers are non functionality
depends on BL.

(vi) **Phase 1**: Evaluate Initial Business Function Ißf
        For each BLi [ where i >= 1] in BL
                : Ißf(i) = Compute Initial_function(BLi )
        If BLi € [ Zßf (x ) ,Ußf (x), Idßf(x) , Prßf(x), Sßf (x), Pßf (x), Cßf (x) ] then
                Ißf(i) = BLi where in BLi is computable within the polynomial time.
// Initial Functions includes{ Zßf (x )= 0 | Ußf (x) = 1 | Ißf(x) = x | Prßfi $^n$(x1, x2, x3,… xn)
        = xi | Sßf (x) = x+1 | Pßf (x) = x-1 | Cßf (x) = x' }
        End if
End loop

(vii) **Phase 2**: Evaluate Composite Business Function Cßf
        For each Ißf(i) [ where i >= 1] in Ißf(i)
                : Cßf(i) = Apply composition over Initial Business function Ißf(i)
                If Cßf (x,y ) = Ißfh ( Ißfg1 (x,y ), Ißfg2 (x,y ) )
                        from defined functions Ißfg1, Ißfg2, Ißfh.
        Where Ißfg1, Ißfg2, Ißfh are generated Initial Business Function from the Phase 1
   Cßf(i) = [Ißfg1, Ißfg2, Ißfh] where in Cßf(i) is computable within the polynomial time.
        End if
End loop

(viii) **Phase 3**: Evaluate Recursive Business Function Rßf
        For each Ißf(i) [ where i >= 1] in Ißf(i)
                : Rßf(i) = Apply recursion over Initial Business function Ißf(i)
                If Rßf (x,0) = Ißfg1 (x) and
                        Rßf (x,y +1)= Ißfh ( Ißfg2 (x,y ), Ißf(x,y ))
                from defined initial business functions Ißfg1, Ißfg2, Ißfh
       Where Ißfg1, Ißfg2, Ißfh are generated Initial Business Function from the Phase 1
    Rßf(i) = [Ißfg1, Ißfg2, Ißfh] where in Rßf(i) is computable within the polynomial time.
        End if
End loop

**(ix)** If (Input_BusinessLogic(BL) ) is identifiable in all the three phases then
BL is totally functionally computable
Otherwise BL is partially computable
End.
```

```
(x) **Phase 1 (a)**
      Evaluate each non functional parameter for each simple BR € BL
      For each NFBRi [where i>=1] in BL
        If  NFBRi  € [AuR(x),AtR(x), Constraints,… ] then
               NFBRi is computable within the polynomial time
            Endif
     End loop

(xi)  **Phase 2 (a):**  Evaluate each non functional parameter for each composite BR € BL
            For each NFBRi  [where i>=1] in BL
                Check for composite BR:
                        CNBR1,CNBR2….CNBRn € NFBR1….NFBRn
            If CNBRi is independent then
                        CNBRi+1   (skip next)
                        If CNBRi depends on other NFBRi….NFBRn then
                                    If BnRi…..BnRn € [AuR(x), AtR(x),constraints…] then
                        CNBRi is computable within the polynomial time
            Endif
Endloop

(xii) **Phase 3 (a):** Evaluate each non functional parameter for each recursive BR € BL
   For each NFBRi [where i>=1] in BL
            Check for recursive Br:
                        If RNBR(x)=NFBR(x) and
                                    Where x= [AuR(x),AtR(x),constraint)
                                    RNBR(S(x+1))=g(x,f(x))
            Endif
Endloop

**(xiii)** If (Input_BusinessLogic(BL) ) is identifiable in all the above  three phases then
BL is totally non functionally computable
Otherwise BL is partially computable

If (Input_BusinessLogic(BL) ) is identifiable in all the six phases then
BL is totally computable
Otherwise BL is partially computable
End.
```

This technique enables us to measure the computational power of business logic since many business logics are bounded with time deadline. Ability to divide the business rule in terms of mathematical recursive function is the key goal of this technique. It is possible to reduce the computational complexity of complex business logics by dividing it into a number of simple business functions and reducing the interaction between them by applying recursion. Before developing core business logics with effective business rules, it is necessary to analyze and model the business logic or business process in terms of computable function to make it worth and bound within the time deadline. For example, in an e-commerce system, the application server may need to process payments, deduct inventory, send order notification and display confirmation once an order is finalized. Each of the stages can be viewed as a business processing task and the status of one stage has influences to the subsequent stages, each with different time deadlines. In order to estimate the complexity of these business tasks within the deadline time, it is necessary to analyze and model these stages in terms of computational power.

Typically these business functions in service composition measures the quality of business logic in terms of functional and non functional aspect, which in turn lead to service computational complexity. The partial and total computability capabilities of business logic can also be estimated in a timely manner at runtime in order to make the service run time manageable. Hence the quality measure of business logic depends on the computational complexity, traceability factors, dependency and reconfigurability of services being invoked.

IV. SECURITY THEOREM OF BUSINESS LOGIC

Business logic is the core and critical part of any business process. It resides in the business logic layer in N-tier E-Business application, which obviously decide success or failure rate of the business process in real time. Perfect modeling of business logic model also handles non functional aspect.

From figure 3

BMM = {A: All Attributes Specified in BMM}
BPM is derived from BMM and has
BPM = {B: All Attributes Specified in BPM}

Obliviously, BPM has at least considerable influence over {A}, or in other words {A} must have at least considerable influence over BPM

$$\therefore BPM = \{A\} \cup \{B\} \in BMM \qquad (1)$$

Similarly, BFM={C: All Attributes Specified in BFM}

$$BFM = \{A\} \cup \{B\} \cup \{C\} \in BPM \qquad (2)$$

BRM={D: All Attributes Specified in BRM}

$$BRM = \{A\} \cup \{B\} \cup \{C\} \cup \{D\} \in BFM \qquad (3)$$

BLM={E: All Attributes Specified in BLM}

$$BLM = \{A\} \cup \{B\} \cup \{C\} \cup \{D\} \cup \{E\} \in BRM \qquad (4)$$

Therefore, BLM also has a least considerable influence over the attributes of BMM, BPM, BFM and BRM. The word attributes includes both functional and nonfunctional attributes. Hence BLM should be modeled, keeping the functional and non functional aspects in the mind. The relationship between these models can be expressed as follows:

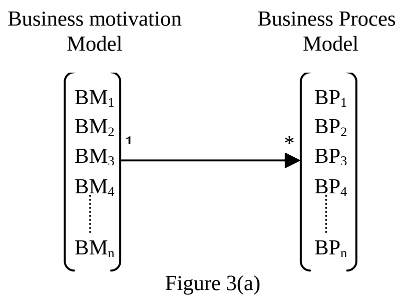


Figure 3(a)

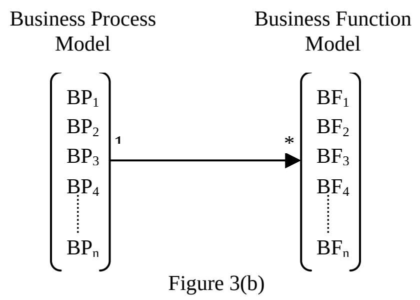


Figure 3(b)

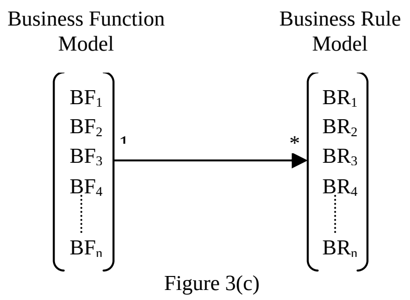


Figure 3(c)

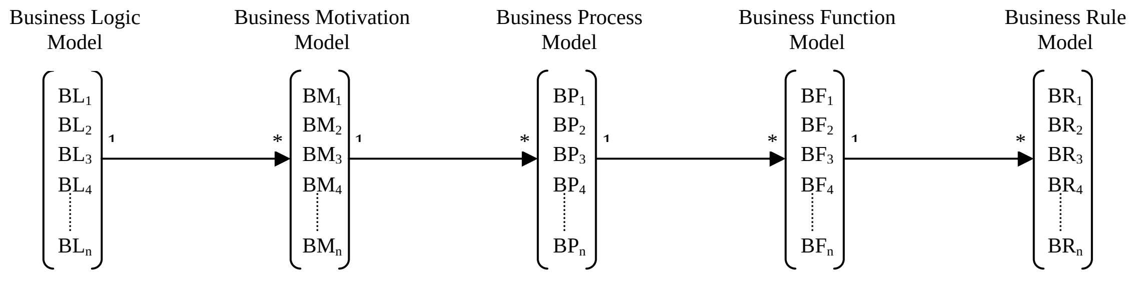


Figure 3(d)

Figure 3: Relation between business models

All these relation are represented for both functional and non functional aspects. In short we can say that business logic is a double edged sword. One deals the significant behavior and future of the organization and other judges how the behavior and future was. This can be shown in calculus as:

For functional aspect

$$BLM_{FRs} = \int_1^c BFM_{FRs} * \int_1^b BPM_{FRs} * \int_1^a BMM_{FRs} * BMM_{FRs} \qquad (5)$$

FRs : Functional Requirements

For non functional aspect

$$BLM_{NFRs} = \int_1^c BFM_{NFRs} * \int_1^b BPM_{NFRs} * \int_1^a BMM_{NFRs} * BMM_{NFRs} \qquad (6)$$

NFRs : Non Functional Requirements
Where a is the number of business process to satisfy the business motivations
b is the number of business function to satisfy the business process
c is the number of business rule to satisfy the business functions

Then, $BLM = BLM_{FRs} + BLM_{NFRs}$ (7)

The equation (7) represents the complete business logic model, which incorporates both functional and non functional aspects. Any partial implementation of the equation (7) may lead to compromise the operational and behavior properties of the organization. The following is the cases for partial implementation of business logic.

Case 1: Considering only the FRs

The FRs describes the organization's functional algorithms which is the principal workflow of the process and procedures to handle information exchange across the organizational boundaries. Negotiating NFRs, will compromise the ability to judge the operation and quality of the system, rather than specific behavior.

Case 2: Considering only the NFRs

NFRs always be contrasted with FRs that defines specific behavior or function. NFRs does not exit without FRs. Considering the NFRs in business logic will improve the execution qualities such as security and usability and evolution qualities such as testability, maintainability, extensibility and scalability of the system.

A. Security Theorem of Business Logic:

If a business function is specified as a compact set on interval [a,b], then the function has both maximum and minimum security density or level on [a,b] . From calculus for a business function f(x) is specified in a closed and bounded interval [a,b]. Then f(x) must attain its maximum and minimum level of security at least once i.e. there exists c and d in [a,b], where c and d is the low level and high level of security respectively and 'a' and 'b' are the minimum and maximum consideration of NFRs,

such that $f(c) \geq f(x) \geq f(d) \qquad \forall x \in [a,b]$ (8)

For a business function f(x), 'r' is non empty set of business rules, then the quality or security aspect 's' of 'r' under the business function f(x) is the set of all quality/security aspects 's' that can be produces by applying NFRs over f(x) to the member of 'r' is a member of nonfunctional business logic. So if 's' is the member of NFRs, then F(x) is also the member of NFRs, under f(s). f(x) is said to be secured iff(if and only if) all the members of 'r' is the member of NFRs is called Security Theorem of business logic

where F(s) is the BFM for both functional and non functional.

The image of the business logic also has the security attributes as shown above. Moreover using the above theorem it is possible to observe the density of the security attributes of the given business logic.

V. Implementation

With the advent of design principles, patterns and techniques the services are well defined to minimize the gap between the business and enterprise. SOA is not a technology rather it is an architectural style to make the availability of resources to the participants using the service which spans across the boundaries of enterprise which fulfills the business and goals of an organization, which can be choreograph these service to composite application and invoke them through standard protocols.

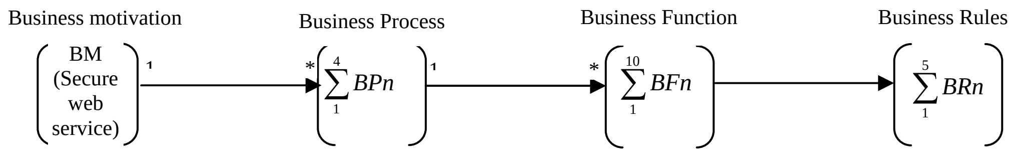
IT professionals and business user are more benefited by combining the business process management and service oriented architecture. Business process management is the building block for service oriented development of application. In the circumstances of natural partners' service oriented architecture and business process must be hand-in-hand to assemble new application for business. In short business process is modeled as a set of individual processing tasks. BPM defines the create process model, process automation, the way to invoke the service and the process flow of the service where as SOA reveals the services to the consumer hiding the information of the service composition and orchestration and flow. Due to this characteristics many companies and IT industries a focusing towards SOA for more strategic and usability. Business Process Management (BPM) is a new IT framework for visualizing, improving and executing new for existing processes.

Due to tight coupling of integration technology and individual business application, a small change in the business process or integration technology may lead to increase the operational cost and hence it affect multiple application beneath this. This tight coupling is a nightmare for the developers to modify the integration technology and interfacing to accommodate the changed business process. The introduction of service is more beneficial, without the service in BPM is complex and brittle. Services reduce the bottleneck of the BPM, the process layer to access the underlying business directly to contaminate with unnecessary details about the technology, APIs and application. Due to above said advantages the SOA and the web service are becoming a popular technology to connect the application both within and across enterprise boundaries. The more popular the more insecure, and security has become an important concern impeding the widespread adoption of SOA. The standardization of business process management and workflow technology has been discussed for more than ten years. Several standardization bodies have proposed specifications for different aspects of business process management.

SOAP and MQ Series doesn't perform default authentication or authorization. To achieve default authentication we propose to implement a new concept in SOA called Expected Clandestine Figure (ECF which may be any hardware serial number) for default authentication of the service consumer. RFID technology smart card has a wider user used in diverse area and application. Usually smart card system works by install a smart card reader at the entrance and connects through network which consist database and system application. Because the system required bigger database that stored staff information, cost of investment will be increase towards the system requirement. This is because usually the staff access can validate through authentication process that compare serial number and data on the card with a serial number stored on database[21]. Similarly by considering the ECF, the service is created from the start along with the usual user name and password, or any bio metric authentication etc. The traditional way of using username and password the consumer may be authorized at the initial stage of binding with the service provider beyond that for each transaction the same username and password is carried out for the preceding service, at this point where the security attacks are happened form the other network. So a mechanism in the service to generate the ECF from the consumer's system is proposed. By ECF, the service provider can be intimated that the authorized user is accessing the service in association with the username and password or any biometric authentication techniques. For each successive request the ECF is used rather than the usual techniques. The consumer or hackers has no aware of ECF which the provider alone can extract the ECF. This caused the attackers to fail in their objective. And also the provider always expects the Expected Machiavellian Figure (EMF, detail of attacker) and if, the server senses the EMF, the appropriate action will be initiated. This paper gives the basic idea to incorporate the said scheme in SOAP protocol for the access control of the web service consumer. In fact several key issues are to be addressed that are important for the deployment of high performance and mission-critical SOAP/XML-based services [22].

Keeping in mind that NFRs and business logic for service modeling gives the built-in security features, the above said ECF scheme was implemented in dot net technologies over the HP BL7600 server with five clients. The analysis of the system is as follows as shown in figure 4:

1. By analyzing the business logic, the business processes are



where $\sum_1^4 BPn$ = {Login, Display, Transaction, Logout}

$\sum_1^{10} BFn$ = {Get user name and password, authentication, identification, retrieve data server, display, get request, check identity, do transaction, log out, nullify all the instance}

Figure 4: Implementation of proposed algorithm

Business rule is a compact statement about an aspect of a business. The rule can be expressed in terms that can be directly related to the business, using simple, unambiguous language that’s accessible to all interested parties. It’s a constraint, in the sense that a business rule lays down what must or must not be the case. At any particular point, it should be possible to determine that the condition implied by the constraints true in a logical sense; if not, remedial action is needed. This interpretation, which might be described as Boolean from a software perspective, is the main reason that the term business logic is so commonly used.

In our case there are five rules:

BR1 – Authenticated user must view his account
BR2 – Only administrator can manage and create the account
BR3 – Only authorized customer can perform transaction to another account
BR4 – Each transaction is performed against the identity
BR5 – All log information viewed only on administrator screen

The fragment of XML version of the business rule

```
<?xml version="1.0"?>
<businessRules>
  <rulesGroup name='bank'>
    <xr:ruleset xmlns:xr ="http://pmd.sf.net/ruleset/1.0.0"
    xmlns="http://pmd.sf.net/ruleset/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pmd.sf.net/ruleset/1.0.0
    http://pmd.sf.net/ruleset_xml_schema.xsd">
      <xr:rules context="customer">
        <validate test="userID=UN"/>
        <validate test="password=PW"/>
        <validate test="ECF=ECF"/>
      </xr:rules>
      <xr:rules context="admin">
        <xr:validate test="adminID=admin"/>
        <xr:validate test="password=PW"/>
        <xr:validate test="ECF=ECF"/>
      </xr:rules>
      <xr:rules context="transaction">
        <xr:validate test="userID=UN"/>
        <xr:validate test="password=PW"/>
        <xr:validate test="ECF=ECF"/>
        <xr:validate test="UAccNo=TAccNo"/>
        <xr:Calculate target="balance"/>
        <xr:value when="balance &gt; amount"></xr:value>
        <xr:value> sum(amount/TAmount/TAccNo)</xr:value>
        </xr:rules>
    </xr:ruleset>
  </rulesGroup>
</businessRules>
```

A. Analysis of the proposed algorithm:

The business logic adapter analyzes and categorized various business functions of the business logic with respect to the current strategy. Business logic consists of number of business functions, which are all computable at reasonable cost within the expected time to achieve the goal. The above business rule can be categorized as simple, composite and recursive business rules

Simple business rules: BR1, BR4
Composite business rules: BR2, BR3
Recursive business rules: BR5

Though this paper deals only the importance of the non functional business properties, we narrowed our analysis only to the non functional aspect,

Phase 1(a):

In our business logic that are 2 simple business rules: NFBR1 is the authentication of the web service consumer against the username and the password, so NFBR1 is computable with the polynomial time. NFBR4 is the identity check of the hardware serial number, NFBR4 is also computable with in the polynomial time.

Phase 2(a)

NFBR2 and NFBR3 are the two composite business rules: NFBR2 is depends on NFBR1 and NFBR4 i.e. NFBR2={NFBR1, NFBR4}, NFBR1 and NFBR4 is computable with in the polynomial time in Phase 1(a), hence NFBR2 is also computable within the polynomial time. NFBR3 is depends on NF BR1 and NFBR4 i.e. NFBR3={NFBR1, NFBR4}, NFBR1 and NFBR4 is computable with in the polynomial time in Phase 1(a), hence NFBR3 is also computable within the polynomial time.

Phase 3(a)

Only NFBR5 is recursive: NFBR5 is recursive against NFBR4 (for each time NFBR5 calls NFBR4 to enter into the log file), NFBR4 is computable within the polynomial time in Phase 1(a), hence NFBR5 is also computable within the polynomial time

The services and its all associated business functions and business rules are represented in this way and their computability factors are measured. The business logic is broke down to business function and business rules, these rules are identifiable in all the three phases of the nonfunctional aspect and it is found that these rules are totally non functionally computable. Hence maximum consideration of non functional aspect may increase security of the web service.

B. Case study

In this section we elaborate the security flaws of one of the largest online retailer, Amazon.com, which is the American-based multinational electronic commerce company launched in 1995, later they concentrated on providing web service, in July 2002, Amazon provides online web services called Amazon Web Service(AWS). In December 2008, it was reported that there is a flaw in the digital signature algorithm i.e. the query (aka REST) request to Amazon SimpleDB over HTTP, this gives way to the attacker to provide the necessary data for the request generated by the genuine user. The AWS signature version 1 signs an HTTP query string as follows [24]:

1. Split the query string based on '&' and '=' characters into a series of key value pairs.
2. Sort these pairs based on the keys.
3. Append the keys and values together, in order to construct one big string(key1+value1+key2+value2+….)
4. Sign that string using HMAC-SHA1 and your secret access key.

Colin spots the problem and explains as follows:

When Amazon invented this signature scheme, they forgot about one of the foremost design principle relating to cryptographic signatures: Collisions are bad, in a well known designed signature system, it should be computationally infeasible to construct two different messages which have the same signature; this prevents substitution attacks where an attacker convinces the key holder to sign a harmless message, and then attaches that signature to a different message. Looking how AWS signature version 1 is computed, it is easy to see how to construct collisions: because there are no delimiters between the keys and values, the signature for “foo=bar” is identical to the signature for “fob=ar”, moreover, the signature for “foo=bar&fooble=baz” is the same as the signature for “foo=barfooblebaz”.

Taking a step back, concerns with this weakness and the way Amazon handles security vulnerability reports in general are:

Vulnerability Reporting

- To report security vulnerabilities Amazon doesn’t provide a visible mechanism for researchers/customers.
- Amazon has no published policy on how they handle vulnerability reports.
- Amazon doesn’t have a dedicated security page, mailing list, RSS feed or pigeon carrier for alerting customers of security issues.

Vulnerability Notification

- Amazon hasn’t notified their customers yet that their SimpleDB and EC2 instances were vulnerable for the past 7.5 months. Obviously the vulnerability itself goes back further than 7.5 months.
- Amazon has not published how customers can determine if the issue was exploited. Clearly, Amazon doesn’t have full visibility of the network between the customer and their data centers.

Secure Software Development

- Amazons’ internal code review process failed to detect a basic cryptography flaw. Note, although the error is pretty basic, errors do happen. The real problem here is the failure of peer review processes to identify the problem prior to going into production. Cryptography routines along with other sensitive security routines should be subject to greater security analysis as clearly the stakes are higher. If they did in fact have this code reviewed for security weaknesses, this is a huge oversight. So this is now fixed, what else is vulnerable? It would be great to learn that the lengthy delay in delivering the fix was due to Amazon have all their source code reviewed for security issues…
- Amazon has made no public commitment regarding how security is factored into their software development lifecycle.

The flaws in this signature were reported to Amazon by Colin via a email to Amazon. Amazon’s security people realized that this also affected EC2(Amazon Elastic Computer Cloud) and Simple Queue Service(SQS). Remembering the software engineering principle an error in the development phase will be an exponential to the next phase of the software development life cycle and the cost to rectify the errors also expensive. It took about 7.5 months to fix the problem; this is a considerable length of time for the simple weakness.

The ability to handle non-functional properties (or Quality of Service attributes) of services in business logic is desirable. Web service based software development is becoming a popular approach. Business logic approach helps to manage the complexity and diversity of products and to reduce lead time and development costs. However, main advantage of web service is reuse and loosely coupling. Web service often causes problem at real time situation, as the non functional requirements are not satisfied even if the functional specification are met. In our case study Amazon neglects the specification of non-functional Quality of Service (QoS) properties such as security, dependability, costs, response time, reliability and scalability. Due to this negotiation of non functional properties Amazon faces a tremendous fall in the ecommerce environment and losses the credibility of the customer. The proposed approach gives the clear understanding of the system to be developed and also provides the feedback as partial computation and fully computation as explained in the previous section.

VI. CONCLUSION

Business models play an important role in expressing the business needs in concrete ways, controlling the business operation work flow, handling run-time exception, managing the business object, consisting of the business logic with respect to the context, etc. As a result, this business logic serves as the driving factor for business systems, which form an essential part of the business processes containing the set of business rules and business functions bound with policies and standards. Further, a Case Study has been illustrated for the proposed business logic model for service computing and the business logic used dynamically handles exceptions during run-time using the cellular pattern for web services exception handling. Subsequently on implementation and analysis of the model, the quality attributes like computability, traceability and decidability have been evaluated for an enhanced performance. Finally, the proposed business logic used in the devised web service computing system varies from the models discussed in the existing literature by not involving any other external resource dependency or technology and has encouraged the arrival of novel QoS parameters such as Computability, Traceability, Decidability, Manageability, etc. As directions of future research, in addition to the exception handling mechanism proposed in the Business Logic Model as discussed above, this model can be extended to prevent violation, that exist frequently in the existing web service computation and composition, such as loss of generative power service provider's business logic.

REFERENCES

[1] Dirk Draheim, Gerald Weber, "From-Oriented Analysis, A New Methodology to Model Based Application", Springer, vol 4(3), pp 346-347, 2005

[2] Ronald G.Ross, "Principles of the Business Rule Approach", Addison Wesley Publisher, ISBN 0-201-78893-4, 2003.

[3] Asuman Dogac, Yildiray Kabak, Tuncay Namli, and Alper Okcan, "Collaborative Business Process Support in eHealth: Integrating IHE Profiles Through ebXML Business Process Specification Language", IEEE Transactions on Information Technology in Biomedicine, vol. 12(6), pp 754-762, 2008.

[4] Saqib Ali, Ben Soh, and Torab Torabi ,"A Novel Approach Toward Integration of Rules Into Business Processes Using An Agent-Oriented Framework" , IEEE Transaction on Industrial Informatics, Vol. 2(3), pp 145-154, 2006.

[5] Luiz Marcio Cysneiros, Julio Cesar Sampaio do Prado Leite and Jaime de Melo Sabat Neto, "A Framework for Integrating Non-Functional Requirements into Conceptual Models" Springer. LNCS, Issue 2068, pp 284-298, 2001.

[6] Finkelstein A, Dowell J, "A Comedy of Errors: the London Ambulance Service Case Study" Proceedings of the Eighth International Workshop on Software Specification and Design, IEEE Computer Society Press, pp 2-5, 1996.

[7] Breitman KK, Leite JCSP, Finkelstein A. "The world's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study" Brazilian Computer Society, pp 13-37, 1999

[8] Wada. H, Suzuki. J and Oba. K "A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture" IEEE Conference on Service Computing, pp 187-195, 2007

[9] Wada. H, Suzuki. J and Oba. K, "A Model-Driven Development Framework for Non-Functional Aspects in Service Oriented Grids" ICAS, IEEE Computer Society, pp 30-38, 2006

[10] S. Paunov, J. Hill, D. C. Schmidt, J. Slaby, and S. Baker, "Domain-Specific Modeling Languages for Configuring and Evaluating Enterprise DRE System Quality of Service". Proceedings of IEEE International symposium and Workshop on the Engineering of Computer Based Systems, pp 198-208, 2006

[11] D. C. Schmidt, "Model-Driven Engineering", IEEE Computer, 39(2), pp 25-31, 2006.

[12] Z. Gu, B. Xu, J. Li, "Inheritance-Aware Document- Driven Service Composition", Proceeding of IEEE International Conference on E-Commerce Technology and on Enterprise Computing, ECommerce, and E-Services, pp. 513-516, 2007.

[13] S.C. Oh, J.W. Yoo, H. Kil, D. Lee, and S. Kumara, "Semantic Web-Service Discovery and Composition Using Flexible Parameter Matching", Proceedings of IEEE International Conference on E-Commerce Technology and on Enterprise Computing, ECommerce, and E-Services, pp. 533-536, 2007.

[14] John Jung, Soundar Kumara, Dongwon Lee, and Seog, "A Web Service Composition Framework Using Integer Programming with Non-Functional Objectives and Constraints" IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pp 347-350, 2008

[15] Asoke K Talukder and Manish Chaitanya, "Architecting Secure Software Systems", Auerbach Publications, 2008.

[16] Asoke K Talukder "Analyzing and Reducing the Attack Surface for a Cloud-ready Application" Indo-US Conference on Cyber Security, Cyber Crime, and Cyber Forensics, National Institute of Technology Karnataka, 2009

[17] G. Sindre and A.L. Opdahl, "Eliciting Security Requirements by Misuse Cases," in Proceedings of 37th Conference on Techniques of Object-Oriented Languages and Systems, TOOLS Pacific 2000, pp. 120–131, 2000

[18] Saqib Ali, Ben Soh, and Torab Torabi ,"A Novel Approach Toward Integration of Rules Into Business Processes Using An Agent-Oriented Framework" , IEEE Transaction on Industrial Informatics, vol. 2(3), pp 145-154, 2006.

[19] Wada. H, Suzuki. J and Oba. K, "A Service-Oriented Design Framework for Secure Network Applications" Computer Software and Applications Conference, IEEE, 2006

[20] Chollet, S.; Lalanda, P.; Bottaro, A. "Transparently Adding Security Properties to Service Orchestration" IEEE, International Conference on Advanced Information Networking and Application, pp 1363-1368, 2008

[21] Muhammad Tarmizi Lockman, Ali Selamat, "Verification and validation communication layer of embedded Smart Card system " IEEE international conference on Elcetronic Design, pp 1-5, 2008

[22] Robert A. van Engelen , "Pushing the SOAP Envelope with Web Services for Scientific Computing", In proceedings of the International Conference on Web Services (ICWS), pp 346–352, 2004.

[23] G.Sindre and A.L.Opdahl, "Eliciting Security Requirements by Misuse Cases", in the proceedings of 37th Conference on Techniques of Object Oriented Language and Systems, TOOLS Pasific 2000, pp 120-131, 2000.

[24] www.cloudsecurity.org

[25] www.harriman-house.com

[26] www.issco.unige.ch

[27] www.bptrends.com