American Journal of Applied Sciences 11 (8): 1241-1249, 2014
ISSN: 1546-9239

doi:10.3844/ajassp.2014.1241.1249 Published Online 11 (8) 2014 (http://www.thescipub.com/ajas.toc)

BUSINESS LOGIC EVALUATION MODEL USING DEPENDENCY ANALYSIS APPROACH FOR SER-VICE INTEGRATION

[1]Thirumaran, M., [2]M. Jannani, [3]P. Dhavachelvan, [4]N. Balaji and [5]M.S. Saleem Basha

[1,2]Department of CSE, Pondicherry Engineering College, Puducherry, India
[3]Department of CSE, Pondicherry University, Puducherry, India
[4,5]Department of Computer Science, Pondicherry University, Puducherry, India

Received 2013-04-12; Revised 2013-07-16; Accepted 2014-05-15

ABSTRACT

In order to extract and integrate the required business logics from the rapidly expanding business services accurately and efficiently, developers must fathom the entire service and must decide on the proper approach to merge them which is a complex and time-consuming task. So integrating the service logics automatically by scrutinizing the dependency relations existing on business rules, functions and parameters is a challenge in the current scenario which is the motivation of this study. In this study, we address this challenge by proposing a comprehensive framework for dynamic service integration which examines the service logics and integrates them dynamically as interoperability between the service logic is attained. The incorporation of Business Logic Evaluation Model in the presented framework employs a novel approach called dependency analysis which ascertains the dependencies among the service logics through Finite State Ma-chine and integrates from the possible dynamic service integration constructs such as union, composition, substitution and reducibility. The implication of this study is to allow enterprises to share and integrate the service logics even without developer's intervention at a much better level. The implementation methodology and evaluation metrics for the proposed framework have also been elucidated.

Keywords: Service Integration, B2B Collaboration, Business Logic Property Evaluation System, Computa-Tional Criteria

1. INTRODUCTION

Service integration has been designed to enable the organizations to attain integration maturity for integration between enterprise applications among partners. But the current market demand does not get satisfied with integrating applications or web services as a whole. It requires integrating service logics in business rule or functional level for different requisite automatically. Integration in this level has two main challenges. First, required logic need to be located and extracted from the whole service. Second, the retrieved logic must be integrated efficiently as interoperability is attained. It is a complex task which needs developers to understand the entire services and identify better way for integration. So the demand is to have a mechanized system to integrate the services dynamically. The framework proposed in this study responds to these challenges in three fold: Firstly, it formats the request and discovers required service logic through rule manager. Secondly, extracted logics are built as a complete web service and FSM is constructed by the Business Logic model to state the logic flow. Thirdly, it ascertains the right way for integration by analyzing dependency between the service logics through FSM. Finally, service logics are integrated using corresponding template and deployed in to server. This allows enterprises to share and integrate the services dynamically without developer's intervention at any stage. Thus this can be used for any IT enterprises in

Corresponding Author: Thirumaran, M., Department of CSE, Pondicherry Engineering College, Puducherry, India

modern service industry to reduce the threshold of development and operation of services. The objective of the proposed framework is to facilitate d-ynamic service integration which facilitates enterprises to share their service logic more sophistically and securely with their business network partners.

2. RELATED WORKS

The works pertinent to the focus of the paper have been elaborated. In order to realize efficient information sharing and interoperability among varying platforms or large-scale systems, (Hui-Fang and Guang-Feng, 2010) have presented an approach to SOA-based service integration which makes the service granularity flexible to change. It is shown that the flexibility and running efficiency of system are improved. Zhang and Ben (2009) have proposed an agent-based Web services integration model and designed the organization structure and interactive pattern of the multi-agents system in this model. Yong (2010) has employed Enterprise Service Bus (ESB) to share the resources between the organizations. Source control management system in the model facilitates enterprises to carve up the resources as stated by the SLA. Asuncion *et al*. (2010) have proposed a model for the separation of business rules from the business process of the integration solution. In an effort to bridge the gap between small and big companies, a Web Service-enabled B2B integration approach for SMEs has been developed by (Yan *et al*., 2008). It provides a feasible and cost-effective solution for SMEs to take part in B2B collaborations by taking advantage of Web services characteristics and lightweight IT infrastructure for SMEs (Yan *et al*., 2008). Rathore and Suman (2011) have proposed a QoS broker based model for dynamic web service composition which solves the problems associated with quality of web service. It also prevents the central repository from malicious service provider to publish wrong information.

3. PROPOSED FRAMEWORK FOR DYNAMIC SERVICE INTEGRATION

Long Proposed framework affords a secure and reliable platform for enterprises to carve up service logics among their partners. It aids in immense way to integrate the service logics embedded in various services to process the requisites. The framework accomplishes these tasks by uniting with Business rule and business logic management system. Business rule management system mainly here ascertains the required part of service logic from the complete service. Business logic management system fosters to do changes in the existing logic and molds it in to different forms. The framework through its various components it integrates the service logics and develops a complete novel service. **Figure 1** exemplifies briefly the working of proposed framework for dynamic service integration. Request analyzer in the top of the framework acquires the service request, slices it into numerous parts and exposes each part into standard format. Service locator discovers correlated services to process each part from service repository. Business Rule Manager manages the set of rules for the services in the service repository and provides powerful search and management tools that allow business users to locate the logic easily. Managing these business rules increases the company's flexibility by allowing tactical changes to be rapidly reflected in the operation of our enterprise applications. Using rule manager, new business rules can be added efficiently and existing business rules can be quickly uncovered from within code and annotated for better control. Tracing business rules ensures the consistency of operations when migrating to a new system and demonstrates compliance with laws, standards and regulations. Source manager locates the part of service logic discovered by rule manager and examines authentication and authorization control of the requestor to access the service through the service contract made between them. If the requestor passes all this preliminary tests, it opens the located service logic in editor and gives full control to business logic analyzer. Business logic analyzer then categorizes the located logic into business rule, function and parameters. Dependency analyzer analyzes the dependent rules, functions and parameters of each located rules and function through Rule Bound and Function Bound Analyzer. Dependency Analyzer analyzes the dependency through Finite State Machine (FSM). FSM simulator in the framework simulates FSM as its states represent rule, function and parameter and its transition represents the flow of the logic. Rule and Function Bound Analysis approach uses this FSM and by tracing transition it establishes the dependent parts in the logic. The determined rules, functions and parameters are extracted separately and new service is developed. Computability Engine then examines the computability criteria of the developed logic through the same FSM. It runs the FSM if it halts within a time limit, Property Evaluator in the framework evaluates the logic with various properties such as computability, completeness, accessibility and configurability which are discussed briefly in next section.

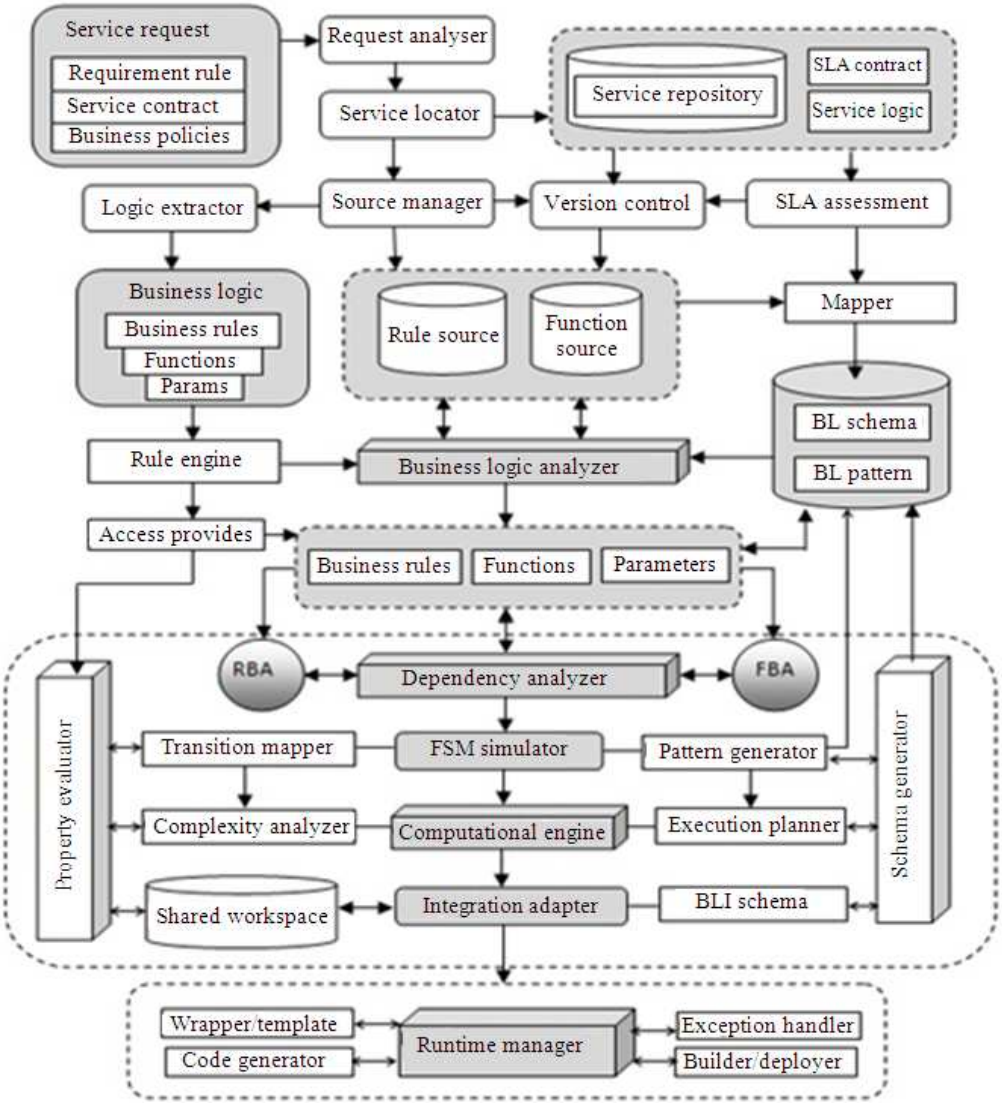

Fig. 1. Change impact analysis framework

If the performance result is good, generated WSDLs are placed in to some common work space and analyze the dependency between them from the given requisite. Integration adapter ascertains the exact construct from the available integration constructs such as union, composition, serialization and reducibility. Then new business logic is developed with the WSDLs in the common work space using corresponding template in right way as fulfilling the client's requirement. Run time manager analyzes the performance of the newly developed logic through evaluation metrics such as timing, hardware counter, synchronization delay, memory allocation and tracing. According to the metrics formulated by performance analyzer, execution planner identifies efficient way to execute the logic and it troubleshoots if any part of the logic is performing poorly. Run time manager through run time Builder/Deployer builds and deploys the integrated logic in to server. Thus this framework integrates the service automatically without any developer's intervention securely and reliably at low cost. Finally integration schema BLI schema is developed holding detailed information about the process and outcome of each component.

4. PROPERTY EVALUATION

At each step of service integration, system evaluates properties such as completeness, configurability, accessibility and computability through various components in the framework to examine proceeding process fulfills the given requirement absolutely. The evaluation process of each property is discussed in detail.

4.1. Computability

Desirable characteristics of business rule or business function to be a computable one must have exact instructions which are clearly defined (i.e., core business logic), finite in length, for the business rule or business function. Thus every computable function must have a finite program (logic) that completely describes how the function is to be computed (i.e., service computing logic). It is possible to compute the business function by just following the instructions; no guessing or special insight is required. If the business function is given a k-tuple x in the domain of f, then after a finite number of discrete steps the function must terminate and produce f(x). Intuitively, the business function proceeds step by step, with a specific rule to cover what to do at each step of the calculation. [Note: If the business function is decomposed into number of sub functions which maps with defined initial function and represented using composition and μ-recursion is said to be Primitive Business Function] Only finitely many steps can be carried out before the value of the function is returned. If the function is given a k-tuple x which is not in the domain of f, then the function might go on forever, never halting, then it is hold by some sensitive exception. Or it might get stuck at some point with some basic types of exception, but it must not pretend to produce a value for f at x. Thus if a value for f(x) is ever found, it must be the correct value and hence, the business function must be totally computable. It is not necessary for the computational engine to distinguish correct outcomes from incorrect ones because the logic of the function is always correct when it produces an outcome. The function must theoretically work for arbitrarily large arguments in order to avoid out of bound or out of range exception. The procedure is required to halt after finitely many steps in order to produce an output, but it may take arbitrarily many steps before halting. No time limitation is assumed. Although the function may use only a finite amount of storage space during a successful computation, there is no bound on the amount of space that is used. It is assumed that additional storage space can be given to the function whenever the function asks for it.

4.2. Completeness

The problem is to prove the business rules are complete and also to show the rules are semantically valid. When the correspondence between the syntax and semantics tighter, we would say the logic is complete. Generally the business logic consists of four major tuples such as rules, functions, parameters and dependency relation which are related as:

$$\exists r[rules\, r] \rightarrow \exists \forall f[\, functions\, f\,] \leftrightarrow$$
$$\exists \forall p[\, parameter(p) \leftrightarrow \exists \forall r[relation\, r]$$

Therefore any semantically valid argument can be captured by formal proof. The choices of rules are to be made for its completeness. It can be easily done when the system is in static phase. When it is done in runtime the conditional variable and iterative variable are also changes. When these variables are modified it brings out bugs in the logic. So these variables must be declared with certain range. The model verifies each rule, function and parameter in the logic is complete and computable within a certain limit.

4.3. Configurability

Configurability focuses on establishing and maintaining consistency of performance over the life cycle. It refers to all activities used to identify, control, ensure the change is being properly implemented and to report changes in the software to others who may need to know of them. This includes all activities related to version control and change control. Configurability in business process is a tool in anticipation of a change in management, here it is assumed that these changes have been identified and should be done in several changes in the sector to continue the business processes that can provide good benefits to the organization. To cope with changes in the structure of information in order to obtain a means of developing and addressing these changes will need a baseline or reference standards which work for the management of these products. Basically every system maintains information such as Security settings, authentication, authorization, logging and other parameters. Whenever changes have been made in the file, system verifies the changes are effective and can be adaptable. In dynamic service computing environment, frequently configuring service resources or devices according to the new requirements sometimes fall into error prone task. Automatically configuring the service with respect to the platform, environment, resources or devices is a mission critical

job which has to be handled effectively. System logic is formed of a set of *code segment* '*cs*' consist of set of system information for configuration and maintenance. Configuration logic is part of system logic points out set of logic related to configurability. It holds set of methods such as connectors, drivers, resource types, credentials, access methods, etc. If any modification done in the system logic, it should be verified that it is not affected the associated business logic and updation is trustworthy. To perusal the modification, initially system logic is expressed in first order logic form. Whenever modification is done, modified logic will be exposed in to FOL and compared with the original FOL. Configurability is meaningful only when original FOL and new FOL are same. Let CL and CL' be configuration logic before and after modification, it verifies FOL (CL) = FOL (CL'). Also it verifies business logic output is same before and after modification. Let BL and BL' be business logics before and after modification, it verifies BL∩BL' = φ. For each modification it verifies code segment and examines the updation is fruitful. Modification is reliable only when resource modified before and after are of same type. i.e., database connection type is allowed to modify only when it is replaced with some other database connection type not with network connection type or some other. Let f(x) be original resource defined in the CL, g(x) be the modified one, it verifies f(x) ∩g(x) = c, some constant. i.e., the modification is allowed in CL only when resource types are same.

4.4. Accessibility

Accessibility is used to describe the degree to which a product, device, service, or environment is available to as many people as possible. Accessibility can be viewed as the "ability to access" and possible benefit of some system or entity. Accessibility is often used to focus on people with disabilities or special needs and their right of access to entities, often through use of assistive technology. Accessibility to business logic provides regulated access to the business resources which business experts and analyst need to perform their duties i.e., to change the policies and rules in the business logic of the service. In order to make changes in the business logic business experts and analyst must follow some common mechanism that permits management to specify what the business expert and the analyst can do i.e., which resources they can access and what operations they can perform on a system. In any organization planning to implement accessibility should consider the abstractions like policies, mechanisms. Accessibility policy should be brief which is highly recommended and set at high level. It must also specify how access is managed and who under what circumstances may access what information. The accessibility policies are enforced through accessibility mechanism and they are direct implementations of formal accessibility policy concepts. Finally translate a user's access request in terms of a structure that a system provides for example, a simple table lookup can be performed to grant or deny access.

The business logic gives detail information flow of the organization so to check the security of the business logic is very important task. The business logic is divided into rules, functions and parameters. A business rule is a statement that gives constrains to some aspect of the business that influence the behavior of the business. It focuses on the policies of the organization. It is essential for an organization in achieving its goals. The business rules cannot be broken down further. Business rules express business policy such as channels, location, logistics, prices and products. A function is a concept used in the organization architecture domain and represents what work is done by that organization, or business role. An organization can be designed as a set of business functions and usually the structure of the organization units within an organization is closely based on the business functions. Some of the functions are get, select, store, compare, return, display, catch, etc. The parameter consists of Access control types such as source, resource and environment; these should be associated with level of accessibility it requires. Initially business logic is sliced into business rule, function and parameter. The decomposed part will be exposed into first order logic and checks accessibility level of each part. Accordingly it allows to access. If it is authorized to modify, for each change verifies the modification is fruitful.

5. METHODOLOGY

The use of Dependency analysis methodology analyzes the dependency relation exist between the service logics though Finite State Machine (FSM). **Figure 2** exemplifies the working of dependency analysis approach in which Business Logic (BL) Analyzer analyzes and decomposes the sited logic in to business rules, functions and parameters. Then it constructs business logic Model (M) as it reflecting the dependency between each business rule with its associated functions and parameters. From the logic model, Finite State Machine is constructed which normally annotates the transition from start state to end state.

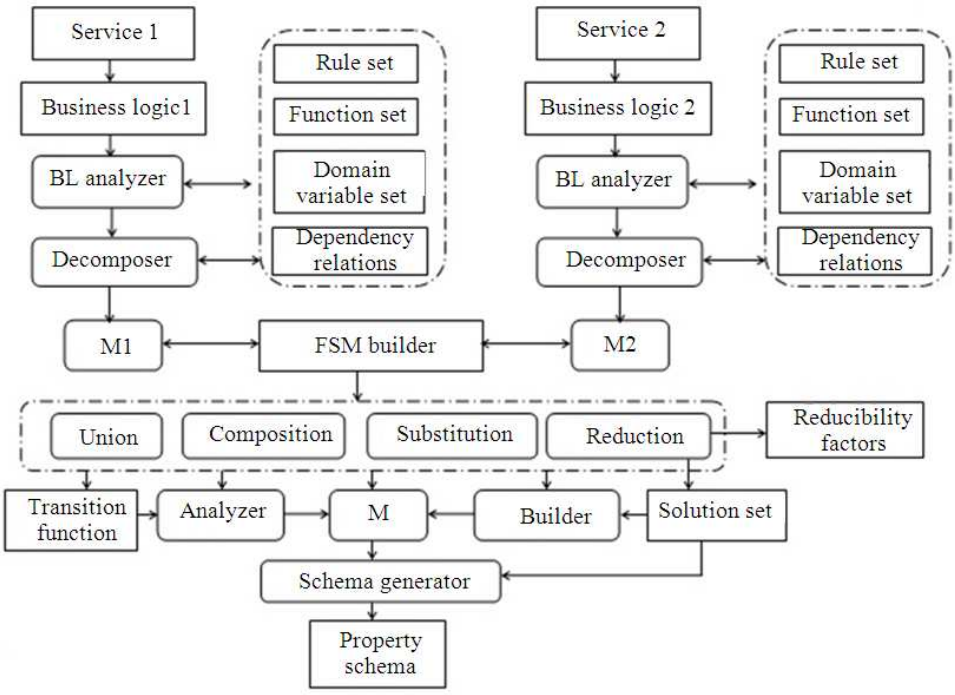

Fig. 2. Dependency analyzer

Here it describes transition from each business rule to business functions, parameters and other business rules. FSM builder analyses the transition states of each FSM and identifies dependency between the two. The mechanism to identify dependency between the FSMs is discussed briefly with theorems and examples for each pattern in this section.

5.1. Union

Theorem: Suppose $M_1 = (Q_1, \Sigma, q_1, \delta_1, T_1, H_1, B)$ and $M_2 = (Q_2, \Sigma, q_2, \delta_2, T_2, H_2, B)$ accept business logics L_1 and L_2 respectively. Let M be an turing machine defined by $M = (Q, \Sigma, q, \delta, T, H, B)$ where Q be the set of States of M, Σbe the input alphabet, q be the start state, δ transition function = $Q \times (T \cup \{B\}) \rightarrow (Q \cup \{H\}) \times (T \cup \{B\}) \times \{R,L,S\}$ is a partial function (that is, possibly undefined at certain points), T be the tape symbol, H be the halt sate, B be the blank symbol, $Q = Q_1 \times Q_2$, $q_0 = (q_1, q_2)$ and the transition function δ is defined by the formula $\delta((p, q), a) = (\delta 1(p, a), \delta(q, a))$ (for any $p \in Q_1$, $q \in Q_2$ and $a \in$ Then If $H = \{(p, q)|p \in H_1$ or $q \in H_2\}$, M accepts the logics $L_1 \cup L_2$.

Proof: Since acceptance by M_1 and M_2 is defined in terms of the functions δ_1^* and δ_2^* respectively and acceptance by M in terms of δ*, we need the formula which holds for any $x \in \Sigma^*$ and any $(p, q) \in Q$ and can be verified easily by using mathematical induction. $\delta^*((p, q), r) = (\delta_1^*(p, r), \delta_2^*(q, r))$. A rule r is accepted by M if and only if $\delta^*((q_1, q_2), r) \in A$. If the set A is defined as $H = \{(p, q)|p \in H_1$ or $q \in H_2\}$, M accepts the logics $L_1 \cup L_2$, this is the same as saying that $\delta_1^*(q_1, r) \in A_1$ or $\delta_2^*(q_2, r) \in A_2$, or in other words, that $r \in L_1 \cup L_2$. Consider the requisite is to develop a service for search by file type and content type. Vistasearch is a book search service which contains business rules to search by file type, date of publication and language. Same way, Flora search is product search service which contains business rules to search by content type, price and brand name. So the required service can be built with the two rules existing in the vista and flora services.

5.2. Composition

Let us consider the composition of functions associated with the business rules of a particular domain and co domain with an example. Let f(), g() and h() be three functions and the logical relation of first two functions are described as f: A→B and g: B→C. This shows the relation between a domain and co domain.

Then for any $x \in A$, f(x) is in the domain of g and it is therefore possible to derive as g(f(x)). More generally, if f: A→B, g: B_1→C and the range of f is a subset of B_1, then g(f(x)) is called composition of g and f and is written h = g o f. For example, the function h is defined by $h(x) = \sin(x^2)$ is g o f, where g(x) = sin(x) and $f(x) = x^2$. The function f o g, on the other hand, is given by the formula $(\sin x)^2$. When you compute g o f(x), take the formula for g(x) and replace every occurrence of x by the formula for f(x). It can also be verified by just tracing the definitions that if f: A→B, g: B→C and h: C→D, then the functions h o (f o g) and (h o f) o g from A to D are equal and they are computed as follows. For $x \in A$, first take f(x); then apply g to that element of B to obtain g(f(x)); then apply h to that element of C to obtain h(g(f(x)). We summarize this property of composition by saying composition is associative. Consider the business rules $r_1 = \{f_1, f_2 \ldots f_n\}$ and $r_2 = \{g_1, g_2 \ldots g_n\}$ then g(f(x)) is called composition of rule r_1 and rule r_2 with their associated functions g and f and is written in general, h = g o f. The function h is the composition function which is defined as h(x) = g(f(x)).

Let us show that if f: A→B and g: B→C are both one-to-one, then so is the composition g o f. To say that g o f is one-to-one means that whenever g o $f(x_1)$ = g o $f(x_2)$, then $x_1 = x_2$. But if $g(f(x_1)) = g(f(x_2))$, then since g is one-to-one, $f(x_1) = f(x_2)$. Therefore, since f is also one-to-one, $x_1 = x_2$. Similarly, if f: A→B and g: B→C is both onto, g o f is also onto. For any $z \in C$, there is an element $y \in B$ with g(y) = z, since g is onto and there is an element $x \in A$ with f(x) = y, since f is onto. Therefore, for any $z \in C$, there is an $x \in A$ with g(f(x)) = g o f(x) = z. Let the requisite is to develop tour reservation service to reserve for accommodation and travels automatically through this. This can be developed simply by composing reservation module of hotel and travels reservation services.

5.3. Substitution

Theorem: Business Logics are closed under substitution.

Proof: Let L be a BL, BL $\subseteq \Sigma^*$ (where Σ^* be the business domain set which includes Rules, Functions, Business variables and Dependency Relation) and for each Rule r_1 in Σ let BL_{Rule1} be a complete BL. Let BL = *L*(G), G = (R, F, P, D) and for each Rule r_1 in Σ let $BL_{Rule1} = BL(G_{Rule1})$, where $G_{Rule1} = (R_{Rule1}, F_{Rule1}, P_{Rule1}, D_{Rule1})$. Assume that $R \cap R_{Rule1} = \varnothing$, for all rules in Σ and $R_{Rule1} \cap R_{Rule2} = \varnothing$, for all $Rule_1 \neq Rule_2$ in Σ.

Construct G'= (R', F', P', D'), where $R' = \cup_{rule1}$ in $_{\Sigma}R_{rule1} \cup R$, $F' = \cup_{function1}$ in $_{\Sigma}F_{function1}$, $P' = \cup_{param1}$ in $_{\Sigma}P_{param1} \cup \{Rule_1 \rightarrow function_{1..x}(Param_{1..y}) | Rule_1 \rightarrow Function_{1..x}$ is in D, $Rule_1(Function_{1..x}) = D_{rule1}$ for each $Rule_1$ in Σ and $Function_1(Param_{1..y}) = D_{function1}$ for each $Function_1$ in $\Sigma\}$ Example: Consider the requisite is to extend an authentication service by adding encryption part into it. Let the security service containing required encryption part. Now this can be added to our service by extracting the required logic as explained in union method. The extracted logic can be substituted in to required place in our service. Now this extended service can be built and deployed.

5.4. Reducibility

If we can establish that one decision logic, L_1, can be reduced to another, L_2, or that having a general solution to L_2 would guarantee a general solution to L_1, then it is reasonable to say informally that L_1 is no harder than L_2. It should then follow that if L_2 is solvable (or equivalently, if L_1 is unsolvable). Reducing one decision Logic to another: If L_1 and L_2 are decision logics, we say L_1 is reducible to L_2 (written $L_1 \leq L_2$) if there is an algorithm procedure that allows us, given an arbitrary instance I of L1, to find an instance F(I) of L_2 so that for every I, the answer for the two instances I and F(I) are the same. Reducing one business rule to another: If r_1 and r_2 are business rules, over the global Rule Sets 1 and 2, respectively, we say that r1 is reducible to r_2, denoted $r_1 \leq r_2$, if there is a Turing-compatible function f: $\Sigma_1^* \rightarrow \Sigma_2^*$ so that for any $x \in \Sigma_1^*$. $x \in L_1$ if and only if $f(x) \in L_2$. For instance a login service requires username, password, date of birth, email id, account no, key as input to check authentication and now the requirement is to have a service that checks username and password only for authentication. The required service can be built by reducing business parameters in the existing service.

6. RESULTS

We have analyzed the performance of service integration by evaluating throughput, co-existence and maximum time required for service integration in various cases. The evaluation results of service integration for Enterprise A and Enterprise B is shown in **Table 1**.

Table 1. Evaluation results of service integration

Enterprise A Service (S1)	Enterprise B Service (S2)	Integrated service	T_{RE} (ms)	T_{SA} (ms)	T_{SC} (ms)	T_{SG} (ms)	T_{SD} (ms)	T_{SI} (ms)
Quick search r1: Search by file type r2: Search by date of pub r3: Search by language	Advance search r1: Search by content type r2: Search by price r3: Search by brand name	Semantic search S1(r1) U S2(r1)	0.954	1.132	0.567	0.674	0.395	3.722
Security service1 r1: Input validation r2: RSA encrypt- Decrypt	Security Service 2 r1: AES Encrypt-Decrypt r2: Confidentiality-SHA	Security service S1(r2)os2(r2)	0.876	1.354	0.678	0.800	0.563	4.271
Online Shopping r1: Manage items r2: calculate amount	Mail Service r1: Mailing r2: Alert msg to mobile	Alert service S1(r2)-> S2(r2) (Calculate amt , if not paid,send alert message)	0.979	0.996	0.623	0.710	0.482	3.790
Registration r1: Get Contact, Edu, Personal info. r2: Submit	Security service1 r1: Input validation r2: RSA encryption	Registration service r3->S2(r1), r3 = R(S1(r1)) S1(r1) is reduced and it gets contact and edu info only	0.785	1.046	0.637	0.659	0.475	3.904
Billing service r1: Manage customer info. r2: Calculate bill r3: Send bill details to customer r4: Pay amount	Banking service r1: Credit and debit from corresponding account. r2: Send info to both	Online billing S1(r4)o s2(r1)	0.8512	1.267	0.598	0.692	0.495	3.903
Travel service r1: Get customer detail r2: Reserve r3: Cancel	Accommodation service r1: Get Customer detail r2: Reserve r3: Cancel	Tour reservation s1(r2) oS2(r2) Tour cancellation S1(r3) o S2(r3)	0.912	1.160	0.620	0.786	0.477	3.955
Login service r1: Get username,pass, cusid, regdate r2: Login		Login service r3->R(r1) r3 gets username and password only to check authentication	0.865	0.899	0.712	0.683	0.491	3.650
Medicine service Gets a disease a name as input r2: Returns list of medicine name	Best doctor service r1; Gets a disease name as input r2: Returns list of specialized doctor	E-Medical S1(r1 &r2) U S2(r2)	0.785	0.963	0.617	0.672	0.483	3.520
Translator r1: Gets a text and language to be translated r2: Returns text in the required language	search service r1: Get search txt r2: Return web page	search service S2(r2)->S1(r2) Searched web page S2(r2) is given as i/p to S1(r2) with lang, it returns web page in corresponding language	0.956	1.175	0.683	0.625	0.492	3.931

7. DISCUSSION

This section analyses the evaluation results of service integration. The analysis of the performance of service integration has been done by evaluating the following. Throughput = Maximum number of integration systems can access a particular service logic in a given period of time = Number of invocations of a service logic/time-taken. Co-existence = Maximum number of systems can access the web service for integration at a time is the total time required for each piece of framework to execute. Service integration time is computed by summing up rule extraction time, service alignment time, service compilation time, schema generation time and service deployment time.

7.1. Service Integration Time

Service Integration time is defined as the time elapsed to accomplish the whole integration process. In other words, it is the total time required for each piece of framework to execute. Service integration time is computed by summing up rule extraction time, service alignment time, service compilation time, schema generation time and service deployment time. *Rule Extraction Time:* Rule Extraction time (TRE) is defined as the time spent by the system to process the request,

locate and extract the business rule. TRE = Time taken to process the Request (TR) + Time taken to Locate the rule (TL). Let n be number of rules and T be the time taken to identify the rule, then TL=n*T. *Service Alignment Time*: Service Alignment time (TSA) is defined as the time taken to retrieve the business logic for the located business rule and build it as a service. TSA= time taken to retrieve the logic (Tr) + Time taken to build it into a service(Tb). Let N be number of lines to be extracted and Te be time taken to extract one line, then Time taken to extract logic of one rule(Tel)=N*Te. Time taken to extract logic of n rule = $Telini$ = 1, TSA = $Telini$ = 1+ Tb. *Service Compilation Time*: Service Compilation time (TSC) is defined as the time required building and deploying each service. Let Time taken to start server=Ts and time taken to create war file for each server = Tw, then Tsc = Ts + $Twini$ = 1. *Schema Generation Time*: Schema Generation time (TSG) is the time taken to identify the pattern to integrate the deployed services and to generate integration schema holding necessary information for integration process. TSG = Time taken to identify required template+ Time taken to generate schema. *Service Deployment Time*: Service Deployment Time (TSD) is the length of time taken to build and deploy the integrated service. Service Integration Time (TSI) = TRE+TSA+TSC+TSG+TSD. Ouyang and Chen (2008) have investigated a problem which minimizes the number of hops of web services while integrating these web services to finish a set of tasks called Minimum Hops of Service Integration Problem. It is proved that, when there are no precedence relationships between the tasks, the decision problem is NP-complete (Ouyang and Chen, 2008). Here we have evaluated the effectiveness of proposed framework and various mechanisms used in it. We have developed set of services and integrated the service logics in various constructs to produce various outcomes. For each case, we have estimated the metrics and the results are shown in **Table 1**.

8. CONCLUSION

In this study we have presented a powerful framework for dynamic service integration which facilitates enterprises to share their service logic more sophistically and securely with their business network partners. The proposed framework integrates the required service logics robotically without developer's intervention at any stage. Business logic model proposed in this study effectively identifies dependency between the service logics and helps to integrate potentially. Also the framework evaluates various properties such as computability, completeness, accessibility and configurability to precede the integration process efficiently. Mathematical methodology and algorithm for property evaluation are discussed in detail. Implementation techniques and metrics to evaluate the paper are presented briefly. Evaluation results are tabulated and pictured graphically. This proves that this would be a standard platform for service providers to share their resources sophistically and securely. However, the use of Finite state Machine makes the framework less suitable for handling concurrent processes which can be overcome by the use of tuning machines. The future work is to consider more properties other than the identified ones and ensure that the logics are interoperable and also to extend the proposed framework to a unified one for service discovery, composition and integration.

9. REFERENCES

Asuncion, C.H., M.E. Iacob and M.J.V. Sinderen, 2010. Towards a flexible service integration through separation of business rules. Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference, (OCC' 10). DOI: 10.1109/EDOC.2010.10

Hui-Fang, D. and X. Guang-Feng, 2010. A study and design of SOA-based service integration for logistics customs-clearance. Int. Symp. Parallel Distrib. Process. Applic. DOI: 10.1109/ISPA.2010.14

Ouyang, W. and M.L. Chen, 2008. An optimal web services integration using greedy strategy. Proceedings of the IEEE Asia-Pacific Services Computing Conference, (SCC' 08). DOI: 10.1109/APSCC.2008.174

Rathore, M. and U. Suman, 2011. A quality of service broker based process model for dynamic web service composition. J. Comput. Sci., 7: 1267-1274. DOI: 10.3844/jcssp.2011.1267.1274

Yan, W.J., P.S. Tan and E.W. Lee, 2008. A web services-enabled B2B integration approach for SMEs. Proceedings of the IEEE International Conference on Industrial Informatics, Jul. 13-16. DOI: 10.1109/INDIN.2008.4618206

Yong, L., 2010. Study on geography information service semantic integration method based on business template. Proceedings of the International Conference on Computer and Communication Technologies in Agriculture Engineering, (TAE' 10). DOI: 10.1109/CCTAE.2010.5544321

Zhang, H. and K. Ben, 2009. Agent-based web services integration model. Proceedings of the 1st International Conference on Information Science and Engineering, (ISE' 09). DOI: 10.1109/ICISE.2009.217