ISSN:0975-9646

Semantic Reasoner Based Change Management Framework for Ev: LCS

M.Thirumaran [1], G.Gayathry@Brendha [2], Subham Soni [3]

[1, 2, 3] Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry-605014,India.

Abstract- **The need for dynamic business environments and SOE (Service Oriented Enterprise) has increased the purpose of using web services to a greater extent. Long term composed services (LCS) are services that prevail for longer time span, thus the requirement for changes in these services become more prevalent. Thus, managing these changes becomes an important task in Web Services. Although, Change Management is a main area of research, maximum exploration on ontology for performing the changes has not yet been concentrated. This paper mainly concentrates on providing a change management framework based on a semantic component called semantic reasoner for top-down changes with the help of an enriched ontology to achieve the following goals: (i) allow the business enterprises to perform the incoming change requests with the help of analyst (ii) determining the feasibility of the requested change before the changes are implemented at service level (iii) diminish the cost and time that is spent on depending the IT professionals for implementing the changes (iv) reduce the risk and errors that may arise at service level after the changes have been implemented.**

Keywords- **Web Services, SOE, LCS, change management framework, semantic reasoner, Ontology.**

1. INTRODUCTION

Web services are a set of loosely coupled self - contained related functionalities that they can be programmatically invoked and published with the help of a web. A web service has three participants: a service provider, a service consumer and a registry. A service provider publishes the description about its services (Web Service Description Language (WSDL) in an registry called UDDI (Universal, Description, Discovery, Interface).A consumer searches this registry to find the appropriate services that matches the requirements using the description in the WSDL. If an appropriate service has been discovered, the consumer requests the service provider for the services with the help of SOAP messages. The services can also be requested with REST messages where the information about the services and their location should be known priorly. The process of finding a suitable service for fulfilling the request is called service Discovery. Service Oriented Architecture (SOA) is a structure that uses the services to provide Enterprise Solutions .These solutions are built by combining the existing services .The process of combining the services is called Service composition.

The services can also be outsourced services. There are two types of service composed services: Long term composed services (LCS) and short term composed services. Short term composed services prevail for short span of time. e.g.: An university may require services to publish its examination results on web, such services are required only for that duration whereas LCS services prevail for long time span. e.g: A travel agency may require providing its services through the web for a longer period of time. Due to requests, complaints and growth in technology it may be required to make changes in these LCS, which can be of two types: top-down and bottom –up[1]. Bottom-up changes occur when the request for the change arises from the outsourcing services. Top-down changes can be achieved only through the LCS owner [2].

But incorporating and managing the changes is not an easy task because there are number of drawbacks in the existing frameworks as well in the services .In the existing techniques, to incorporate even a small change in the existing LCS, the IT professionals were summoned which lead to the increase in cost and time for the enterprise owners. To perform the changes by the analyst, he should be educated about the details of the services. But the WSDL description of web services are syntactical and do not explicitly define their functionality, on the other hand semantic web services that incorporate ontology for the services provide information about the functionality of the services. Further, in the existing systems there are no techniques that will tell about the services that have to be considered before and after making the changes. Thus, all these pit falls have led to a question :” How to build a framework that will soothe the burden on the analyst and assist him in implementing the changes ?”. Hence, this research mainly concentrates on top-down changes with the help of Semantic web services and an enriched ontology, so that the exact context ,requirements etc. for the change request can be identified.

Ontology is a form of knowledge representation that describes the concepts of a service .It is also used to determine the relationship, types and properties of the service concepts, which is used to classify the web services within a domain. Therefore, we provide a change management framework which is semantically driven to furnish the required information from ontology to satisfy the requested change. Before the changes are implemented at the service level, a prior knowledge about the feasibility of the requested change is provided by a special component called semantic reasoner that uses ontology. Since the execution of the requested change is verified in advance, errors that may arise in the service level due to the implementation of the requested change are prevented. Along with this, to perform the changes successfully, there arises a need to maintain the state information about the services until the process of enacting the changes has been completed. Therefore, the semantic reasoner is also assisted by the Finite State Machine (FSM) graph to maintain the state information of services based on the context of the change request.

In Section 2, we discuss the factors in the related works that motivated us to design our Semantic change Management Framework. Section 3 provides us a detailed description on our Semantic Change Management framework. The scenario that will be used for explaining our framework is discussed in Section 4. Following this the algorithms in our framework is provided in Section 5. Section 6 deals with the results obtained on various LCS sets. We then brief about the contributions made by our framework in Section 7. Finally, we provide the conclusion and future enhancements of our framework in Section 8.

2. RELATED WORKS

2.1. Related works on Change Management

The change management framework [2] discussed by Akram et al combines the ordinary and re-configurable petrinets to deal with the incoming bottom up change requests. Algorithms called change detection, change management and change reaction algorithms are executed on these petrinets to maintain the workflow of the services. The changes implemented are not verified and further the construction of petrinets becomes challenging when the number of services to be included increases. The same author in the architecture described in [3] uses request brokers for performing the requested change .The ontology used is a domain ontology which is used by the request brokers to decompose the request, select the appropriate services, invoke the selected services, perform the changes and finally to provide the results to the user. Allocating an instance of the broker to each user becomes an overhead in this architecture. The change management framework by Xumin Liu [4] deals with top down changes .The change to be made is first observed in the schema graph, a graph derived from the ontological representation. After successful verification the change is implemented at the instance level. Although semantics has been considered, it deals only with single requests at a time and the essence of ontology used for the purpose of change management is also very limited. Xumin Liu along with Bouguettaya try to automate the process of making top down changes by proposing a change management framework which has two components called change model and change reaction. The work of change model is to specify the requested top-down changes, while the goal of change reaction is to enact the changes. Dimitris Apostolou et al[5] present an ontology-based approach where systematic response of e-Government systems is obtained to by applying formal methods. They have claimed that such a synthesis of systematic response to changes with knowledge to deal with them has a positive impact on the change management process.The Table 1 discusses the advantages of our framework with the existing Change management framework. The above change management works require IT professionals to perform the requested change. Due to this the cost and time spent for these people increases. Our work aims in creating an environment to make the changes by the analyst itself without the involvement of the IT people.

Further, the possibility of making is a successful change is not verified before it is being implemented at the service level. Therefore, our framework uses a parsing technique to determine the possibility of making the change so that the burden at the analyst side is reduced. Finally, the observed change management frameworks use petrinets which has many disadvantages like petrinets can be represented only as tasks and are useful for change management scenario where the changes are made by the IT developers

.

TABLE 1

Comparison of Existing Change Management Frameworks

Frameworks	**Memory usage**	**Prior Determination for Implementing changes**	**Automatic verification and validation**	**Probability to reach correct states**	**Professionals involved in enacting the changes**	**Consideration of context for enacting the change**	**Information provided in the ontology**
Existing Frameworks	High	No	No	Low	IT developers	No	Domain
Semantic Reasoner Based Change Management Framework	Low	Yes [technique: Predictive parsing]	Yes [technique: FSM]	High	Analyst	Yes [technique: FSM and ontology representing functionality of services]	Functionality of the services.

But in our framework we use FSM which are designed in terms of states and actions, as result all the services are represented in terms of states so that the exact state in which the requested change has to be incorporated can be easily known by the analyst. Since petrinets are complex structures they require higher memory capacity for its storage due to which there is high degree for reaching undesirable states.

2.2. Related works on web services

Paliwal et al [6] initially uses the domain information in the ontology to categorise their services. However, within a domain there may be number of sub-domains, to perform clustering the services based on the sub-domains the author uses a hierarchical clustering algorithm. To discover the closest services based on the user's request the system uses a technique called Latent Semantic indexing. Though the closely related services are extracted, the extracted service differs in the invocation order of the existing services. Therefore, in the service discovery model [7] the descriptions about the services were stored in OWL-S. Due to this, appropriate services that matched the user request were selected. However, this model returned multiple requests in which some were not relevant as the context of the request was taken into consideration. To overcome this problem, the distance between the concepts in the ontology was determined by a tree structure called semantic distance and semantic distance matchmaking algorithm [8]. However, the issue was in the time spent on creating this concept tree as it was a complex task.

The issue discussed in [7] was also addressed by Mastroiannai [9] which used a P2P framework to place the descriptors of closely related service thereby reducing the cost as the search time will reduced. Further, the discovery systems discussed so far did not provide solution to cases of relocation of services, whereas statistical values like co-occurrence were considered to locate the services after relocation by the P2P framework. In a similar manner SMARTSPACE [10], a middleware assisted by the algorithms smart map and smart cluster was capable of making efficient retrieval by reducing the search space on which the request from the user is placed. Although the techniques in [9] and [10] performed efficient discovery, they could not provide solutions even if the already solved requests were provided in a different context.

The author Parejo proposed an Qos-Gasp algorithm [11] with path re-linking to compose services based on the Qos value of the services under composition. Although this algorithm provided low cost and less execution time these advantages cannot be promised as the algorithm was not tested using real time datasets. Futher as this system did not include semantic information it could not compose services based on the context of the user. Thus the automation system [12] uses a co-ordination engine that performs service composition based on user context. The user information is stored in the form of graph which is constructed from the ontology. Based on this information and the context of the user the co-ordination engine will compose the services. But this co-ordination engine can only compose services for a single request at a time. The idea of using AI plans for composition process was found to be more effective in service composition by using the OOM algorithms as discussed by the author Hatzi [13] who used these algorithms to solve the AI plans. These plans used the semantic information from the ontology that consisted of service descriptions. The major issue faced by this work was during the creation of AI plans as these plans are difficult to be formulated. The work discussed using FPPN (fuzzy predicate petrinet) [11] determines composite services according to the user requirements and behavioural context of the user based on the fuzzy semantics and service information which are represented as a set of horn clauses.

From the discussed techniques we can conclude that the scope for creating an enriched ontology set for service discovery and composition has little consequent effects, this is because the ontology for them can be represented only at the data level, whereas in change management ontology can be created for business functionalities. Thus the possibility of creating an enriched ontology is found to be noteworthy in change management. Therefore, we aim in creating an ontology that also includes the functionality of the web services. Though the above systems used different techniques like fuzzy, AI plans etc. to maintain the context information, all of them had many pitfalls like AI plans are generally very complicate to create and solve; similarly, the fuzzy techniques are not usually discrete, they do not provide definitive answers and they are very difficult to program. As a result, there will be high dependency on the IT professionals if the above techniques are to be used in our framework Therefore, we go for finite state machine to maintain the context information of the services because with the help of FSM, we can represent the services at rule level, policy level etc. so that changes can be made by the analyst easily without the assistance of the IT people.

Similarly some methodologies aimed in providing services based on the Qos because some enterprises are concerned only about providing services that would not violate their policies. Such a mechanism was discussed by Szu-Yin Lin [14] whose goal was to find services that matched with the user requirements. In cases where multiple services had same number of Qos, such services were ranked and the service with highest rank was provided as a solution to the user. In the [14] and [17] the Qos is considered only for service discovery and composition. However, in our work we use these Qos for change management by taking into consideration the semantic information provided for making the changes.

3. SEMATIC CHANGE MANAGEMENT FRAMEWORK

In this section, we will be discussing about the workflow for our change management framework and also the components of our proposed work.

3.1 Architecture of Semantic Change Management Framework

The architecture (Fig 1) will brief us about the components in our framework. The framework is made up of three main components: Ontology Repository; Change Evaluation; Storage Subsystem.

3.1.1Ontology repository

The Ontology Repository consist the detailed information regarding the logic set, dependency set, relationship set ,constraints ,QoS Sets and operator sets of each LCS ,thus forming an enriched Ontology. This repository also provides input to various other components of this framework. The logic set consists of the rules, policies, function, parameters etc. of each LCS. The call flow, I/O and data flow dependencies are represented as dependency sets. The constraints include sequential, parallel and concurrent constraints .The QoS set may correspond to factors like security, cost and performance of the LCS. The operator sets determine the adding, removing, modifying or updating a corresponding LCS for implementing the change. This component will provide the required information to the semantic reasoner for performing the predictive reasoning process.

3.1.2. Storage subsystem

The Storage Subsystem contains a registry which is holds the LCS set, WSDL and the grammar sets for each LCS. This entire Subsystem acts as an input for the request analyser and domain analyser stages. If any major changes are executed in a LCS, it may lead to its versioning, the input for performing such type of versioning is also providing by the registry. The versioning of the LCS are also correspondingly recorded in the Change Audit Log.

3.1.3 .Change evaluation

Based on the request that emerged from the user's or from a higher level, a Change Query (Change expression) is created by the analyst. This Query is then analysed by the change analyser. The analysed Query is then provided as an input to the three main subcomponents of the framework, the subcomponents include: a. Query Processor ; b. Sematic Reasoner ; c. Predictive Parser.

a) *Query Processor*-The analysed query is first processed by the query processor which fetches the appropriate grammar for the Change Query from the current and global LCS grammar sets. These grammar sets are obtained from the respective global and local semantic sets that are stored in the storage repository. The grammar sets will be furnished with information like the services, the operations that constitute the services, rules and policies that are bonded with them.

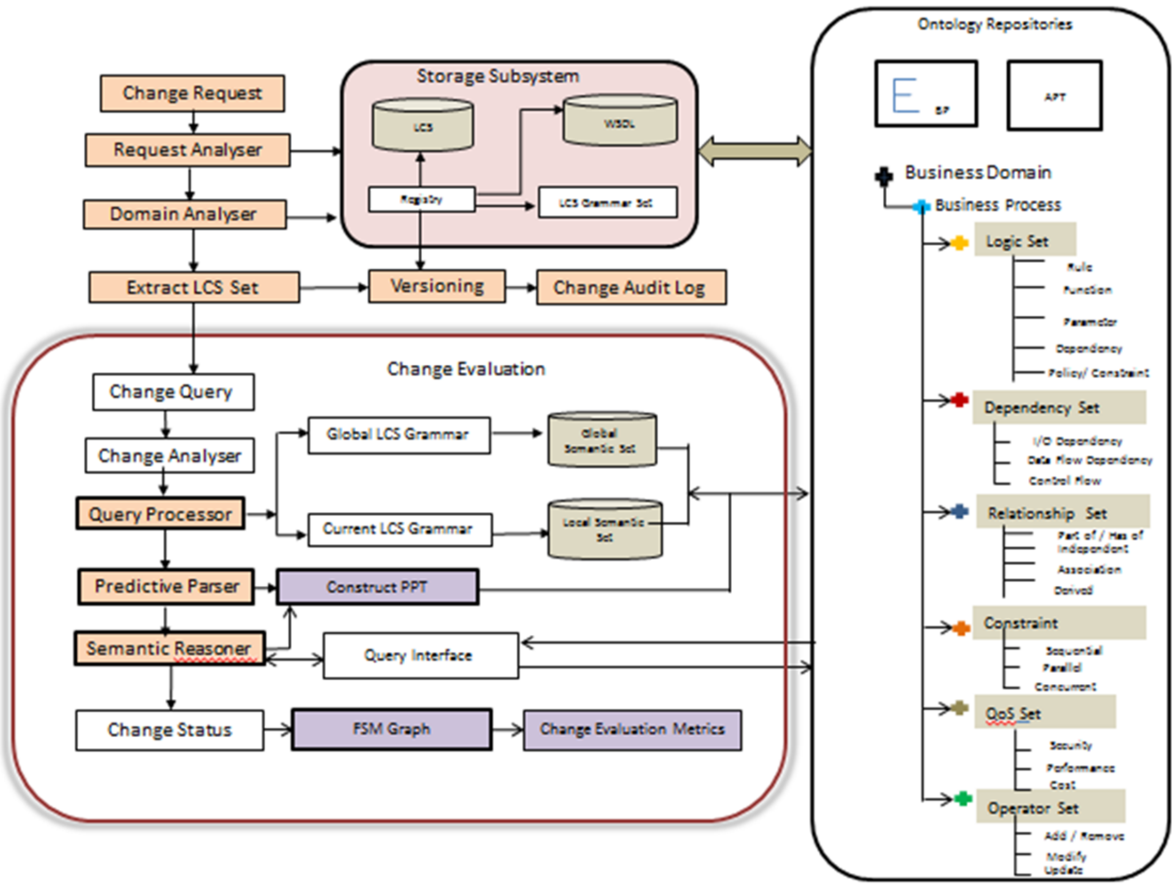


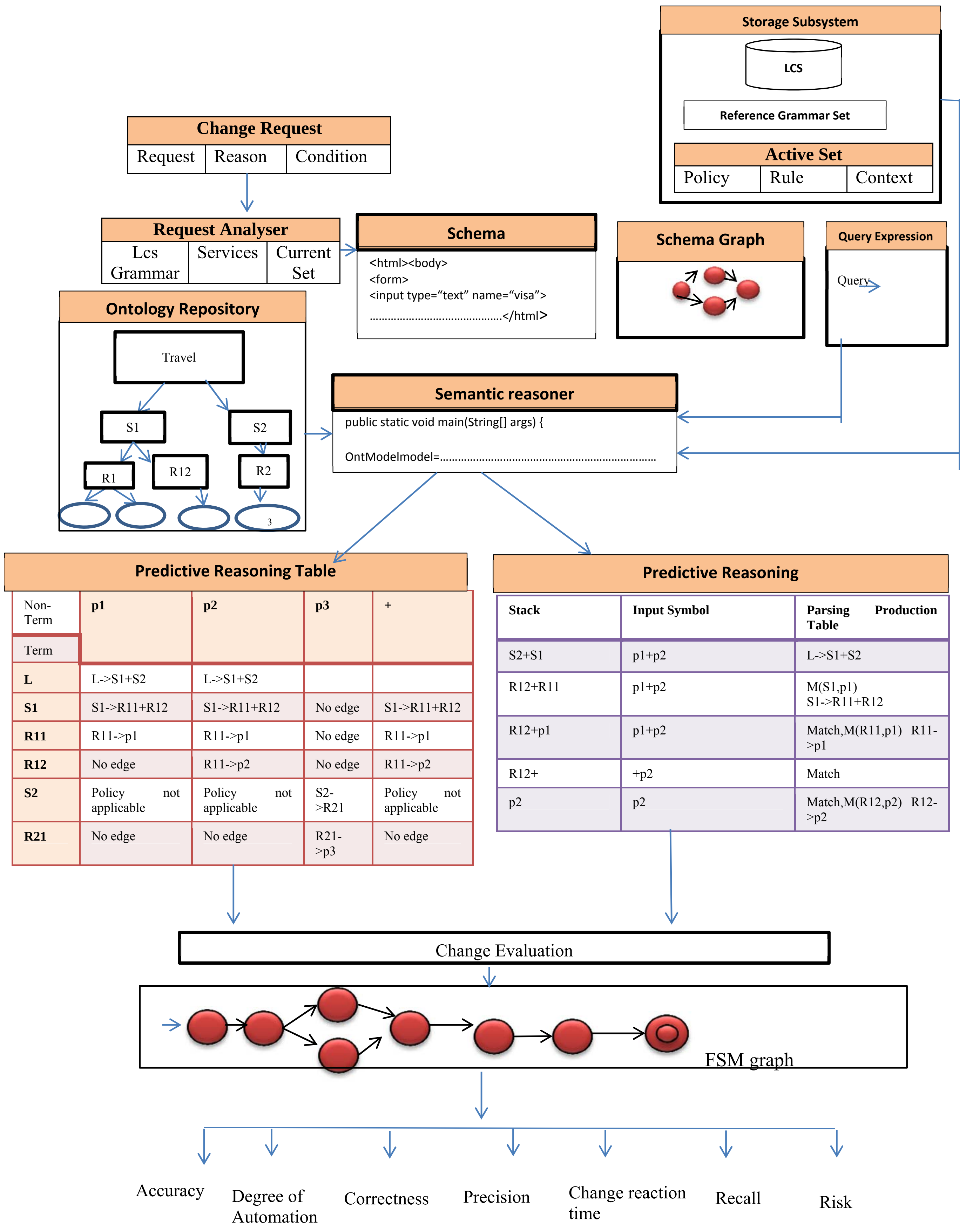
Fig 1. Semantic Change Management Framework.

b*) Predictive Parser*-The output of the Query Processor is used by the Predictive Parser to construct a Predictive Parsing Table. The construction of the Predictive parsing table is a dynamic process comprising of Prequel and Sequel for the grammar. The ontology repository helps to compute the Prequel and Sequel of the grammar sets. The Prequel and Sequel provide information about the resources (services, rules or policy) that should be considered before and after making the change.

c) *Semantic Reasoner*-The Semantic Reasoner is responsible for carrying out the predictive reasoning process, it is a process that utilises the predictive parsing table to justify the implementation feasibility for the change expression. The results of the semantic reasoner are sent to the analyst, the analyst then decides whether to implement the requested change or not. The semantic reasoners will also enlighten the analyst about the services that will be affected by the change so that the analyst can make decision on implementing the change so that he the relationship among the services will not be disturbed.

After the successful implementation of the change request, an FSM graph is formulated to preserve the dependencies and also to support concurrency. The FSM graph is a pictorial representation of the LCS after change which also assists the analyst to conclude on executing the requested change. Following this, the change evaluation metrics such as accuracy, degree of automation, correctness, change reaction time; change evaluation, risk, semantic correctness and level of knowledge gained etc. are calculated.

Fig 2. Workflow for SWCM Framework



```
<html><body>
<form>
<input type="text" name="visa">
........................................</html>
```

```
public static void main(String[] args) {

OntModelmodel=.......................................................................
```

Predictive Reasoning Table

Non-Term / Term	p1	p2	p3	+
L	L->S1+S2	L->S1+S2		
S1	S1->R11+R12	S1->R11+R12	No edge	S1->R11+R12
R11	R11->p1	R11->p1	No edge	R11->p1
R12	No edge	R11->p2	No edge	R11->p2
S2	Policy not applicable	Policy not applicable	S2->R21	Policy not applicable
R21	No edge	No edge	R21->p3	No edge

Predictive Reasoning

Stack	Input Symbol	Parsing Table Production
S2+S1	p1+p2	L->S1+S2
R12+R11	p1+p2	M(S1,p1) S1->R11+R12
R12+p1	p1+p2	Match,M(R11,p1) R11->p1
R12+	+p2	Match
p2	p2	Match,M(R12,p2) R12->p2

3.2 Overview of the Framework

The workflow (Fig.2) will elaborate the change management process that is being followed by our framework. Initially, the request for making the change will be received by the framework, which is then analysed by the request analyser. The request analyser will extract the appropriate LCS and the grammar sets that will be relevant to the received change request will be extracted from the storage subsystem. Following this, the schema and the schema graph for the identified LCS are extracted. The purpose for extracting the schema and schema graph is to provide the knowledge about the relationship between the existing services of the extracted LCS.

After making a thorough study on the schema and the schema graph, the analyst will compose a change expression (query) based on the received request. The change expression is provided as input to the semantic reasoner. The semantic reasoner is responsible for predicting the possibility of implementing the change expression with the help of the predictive reasoning table and predictive reasoning process. The predictive reasoning table is created dynamically based on the Prequel and Sequel. Prequel for a service will determine the services or rules that have to be executed before it. Similarly, Sequel of a service will provide the information about the services or rules that has to be executed after it. Following the predictive reasoning table, the predictive reasoning process will make the decision on the feasibility of the change expression.

If the change is found to be feasible, the change can be implemented at the service level. During this process, the state information of the corresponding services is maintained by the FSM graph until the change management process is completed. The FSM graph will educate the analyst about the state of each service in a LCS at any stage of the change enactment process. Ensuing the successful implementation of the requested change, the incorporated change is evaluated with the metrics like accuracy, degree of automation, precision, recall etc.

Thus, our framework with the help of the semantic reasoner component helps the analyst to decide on enacting a requested change. Further, our framework will also specify about the services that may be affected due to the requested change, so that the analyst will consider necessary steps that has to be taken to maintain the consistent relationship between the services.

4. SCENARIO FOR CHANGE MANAGEMENT FRAMEWORK

Let us consider that Mr. John is running a travel agency which is used for booking air tickets and hotel rooms for its customers by providing their personal information. For security purposes, the agency converts the customer details to a hashed form and then sends it for booking the flight information. After the tickets have been booked, the customer details like date of journey etc. are sent in an encrypted form to the hotel service where the rooms for the customer are booked. Finally, the travel agency sends all the booked details to the customer and the system waits for the payment. The customer provides all his card details and makes a successful payment. Since the service provided by the travel agency is on the web, the payment details are encrypted to avoid any hacking by the intruders on the web.

As the cyber threats are increasing comparatively along with the technology, the analyst finds that both confidentiality and integrity are not preserved at each stage. Therefore, he decides to combine the policies p1 and p2 and provide them at each stage of the booking process so that both confidentiality and integrity are provided to the customers who use this travel agency.

5. EXPERIMENTATION METHODOLOGY

The methodology that is followed by the predictive parsing table and predictive reasoning process will be discussed in this section along with our travel scenario.

5.1. Predictive Reasoning Table

The Predictive Reasoning Table can be computed only after determining the Prequel and Sequel functions. The results of these functions help us in filling the Reasoning Table. The Prequel and Sequel are computed from the LCS grammar. For each LCS, corresponding grammar notations based on rules, services, policy etc. are created and stored in the repository. Non-terminals and terminals for our methodology will depend on the context of the incoming request. For our scenario, since our request is to combine two policies, the terminal symbols will be the policies present the LCS grammar indicated in Fig 3.

Once the grammar for a LCS is extracted, we perform predictive parsing which will parse the change expression to determine whether the requested change is possible for implementation. The analyst will perform the changes only if it found to be feasible. To parse a change expression, the Prequel and Sequel (section 4.2 and 4.3) of the LCS has to be determined. Succeeding the computations of Prequel and Sequel, a predictive parsing table are composed which will help in predictive reasoning.

GRAMMAR	EXPLANATION
L->S1+S2	S1→Airline
S1->R11+R12	S2→Hotel
R11->P1	R11→get customer details
R12->P2	R12→book air ticket
S2->R21	R21→get date of journey
R21->P3	P1→customer details in hash code
	P2→journey details in encrypted form
	P3→payment details in encrypted form

Fig 3. Grammar set for Travel Scenario

5.1.1 Computing Prequel (α)

Prequel of a service is those services or rules or policy that has to be executed before the service that has been taken into consideration.

Therefore, Prequel (α) α is the set of services (non-terminal) symbols and also for the composition operators present in the grammar, then its Prequel can be calculated from the following steps until there are no more symbols to add.

1. If α is a non-terminal, from the ontology we determine if any other terminal say β is dependent onα. If such relationship exists (β→α) then the Prequel (α) will be β.
2. If α is a non-terminal, from the ontology we determine if any other non-terminal say β is dependent onα. If such relationship (β→α) exists, we determine the terminals that belong to β and add them to the Prequel (α)
3. If the encountered symbol α is any of the composition operator, then the terminal that is present prior to it is determined and step 1 is followed.

With the help of the above mentioned rules, the Prequel for the grammar symbols of our travel agency scenario will result in as shown in Fig 4:

SYMBOL	PREQUEL
F(L)	F(L)=F(S1)=F(R11)={p1,p2}
F(R12)	p2
F(S2)	F(S2)=F(R21)=p3

Fig 4. Prequel for Travel Scenario

5.1.2. Computing Sequel (α)

The Sequel for a service will forecast about the services or rules that should follow or will be affected by the change.

Sequel (α) for a LCS can be obtained from the following steps,

1. If α is a non-terminal, from the ontology we determine if α is dependent on any other any other terminal say β. If such relationship exists (α→β) then the Sequel (α) will be β.
2. If α is a non-terminal, from the ontology we determine if α is dependent on any other non-terminal say β. If such relationship (α→β) exists, we determine the terminals that belong to β and add them to the Sequel (α).

3. If the encountered symbol α is any of the composition Fig 5. Sequel for Travel Scenario

SYMBOL	SEQUEL
S(S1)	{+}
F(R11)	{+}
F(R12)	{+}

operator, then add them to the Sequel (α).

Fig 5. Sequel for Travel Scenario

By applying the above rules, we can determine the Sequel (Fig 5) for our case study discussed in section 4.

Once the Prequel and Sequel of the respective LCS grammar set (LG)has been found ,we can generate a Predictive Parsing Table for that grammar using the algorithm (Fig 5).This algorithm takes a LCS grammar set which consists of rules, services, policy, relationships, constraints etc. and produces a Reasoning table as the output. Find Prequel and Sequel of all terminals of the production in LCS grammar. For Every service composition of the form A→γ present in the LCS

```
Algorithm Predictive Reasoning Table (LG)
Input: LCS grammar set (LG) consisting of rules, services, business policy, relationships, constraints etc
Output: A Reasoning Table (R).
Begin
For each production A→ γ of LG, continue
For each terminal a in Prequel (α) add A→ γ to R [A, α]
For each terminal b in Sequel (A).
End If
End For
End For
End For
End For
```

Fig 6. Algorithm for Predictive Reasoning Table

grammar, if there are any terminals “α “present in the Prequel (A) then the respective composition is added to the Reasoning table R [A, α] .

Similarly, the Predictive reasoning table for the travel agency scenario (Fig.7) will be computed based on the algorithm present in Fig 6.

Non-Terminals	**p1**	**p2**	**p3**	+
Terminals				
L	L->S1+S2	L->S1+S2		
S1	S1->R11+R12	S1->R11+R12	No edge	S1->R11+R12
R11	R11->p1	R11->p1	No edge	R11->p1
R12	No edge	R11->p2	No edge	R11->p2
S2	Policy not applicable	Policy not applicable	S2->R21	Policy not applicable
R21	No edge	No edge	R21->p3	No edge

Fig 7: Predictive Reasoning Table for Travel Scenario

5.2. Algorithm Predictive Reasoning (E, R)

On successful construction of the predictive parsing table, the change expression has to be parsed to determine its feasibility. This is done with the algorithm (Fig 8) which works by taking a change expression (E) from the analyst and an appropriate reasoning table (R) as input. The goal of this algorithm is to find whether the given change expression can be reached .i.e. reaches the $ symbol. Initially, the input is made to point the first symbol of the query expression. We assume X (service, rule or policy) to be the symbol in the top of the stack and "Y" be the symbol pointed by the input. The symbol X is popped from the stack when X is a terminal, $ or when R[X, a] leads to any service composition.

```
Algorithm Predictive Reasoning (E, R)
Input: A Query Expression (E) and a Reasoning Table(R) for the LCS grammar (LG).
Output: If E is in L (LG),a leftmost derivation of otherwise error(e).
Begin
Set input to point to the first symbol of E$
Repeat
Let X(rule/policy/service/relationship/constraint)be the top stack symbol and a the symbol pointed to by input
If (X is a terminal or $) then
pop X from the stack and advance input.
Else If (R[X , a]=X→$S_k$,$S_{k-1}$………..S)then begin
pop X from the stack
push $S_k$,$S_{k-1}$,…………,$S_1$ onto the stack ,with $S_1$ on top
output the production X→$S_1$,$S_2$,…………$S_k$
End
Else e().
Until X=$
End.
```

Fig 8: Algorithm Predictive Reasoning (E, R)

For our change request which is combining the policies p1 and p2. The predictive reasoning process is enforced on the change expression (p1 + p2) to determine the possibility of its execution with the help of the algorithm discussed in Fig 9. As a result, the stack operations for it have been represented in Fig 8.

Stack	**Input Symbol**	**Predictive Reasoning Table**
$ S2+S1	p1+p2	L->S1+S2
$R12+R11	p1+p2	R(S1,p1) S1->R11+R12
$R12+p1	p1+p2	Match, R(R11,p1) R11->p1
$R12+	+p2	Match
$p2	p2	Match, R(R12,p2) R12->p2

Fig 9: Predictive Reasoning Process for Travel Scenario

Initially, a $ is pushed onto the stack and the input pointer is made to point the first symbol (p1) of the change request. Since R[S1,p1] leads to a service composition R11+R12, it is popped out from the stack. Now the top element on the stack will be R11 which is again popped from the stack and replaced by p1 as R11 leads to a service composition R11→p1.The stack top and the first symbol of the change expression will be the same and thus they are popped from the stack and the input pointer is made to point the next symbol in the change expression. Similarly, the algorithm is followed till the $ symbol on the stack is reached. On reaching the $ symbol, we conclude that the change expression can be implemented successfully. Following the Predictive reasoning process, the analyst proceeds to implement the services at the service level.

6. RESULT ANALYSIS

Our Change Management Framework is developed with help of software development environments like Netbeans and a tool called Protégé. Protégé is an ontology tool which has a graphical interface helping us to create ontology. Once the ontology has been created, a corresponding owl file for the created ontology will be generated, which is used for inferring information from the ontology. Similarly, Netbeans is an IDE that provides an environment to develop and deploy the services required for our LCS.The information from the ontology can be inferred with the Jena API, which is a Java library that can be added in the Netbeans environment to perform the information retrieval from the ontology.

Based on the incoming change request, the analyst will choose an appropriate LCS to make the changes. The LCS size is the number of services that build up the LCS. For instance: in our running example, the LCS considered for parsing is made up of two services. Therefore, the size of the LCS is two. The Table 2 calibrated by considering the various LCS size. The highlighted results in the Table 2 are the results obtained for our running example.

Table 2

Result Analysis for Various LCS sizes during Successful Cases

No of change requests	LCS Size	Effect of Change	Accuracy	Correctness	Change Reaction Time	Level of Knowledge Gained	Risk
1	2	Success	99.9%	99%	2sec	3	0.1%
2	5	Success	99.9%	99%	7sec	5	0.1%
5	3	Success	99.9%	98.9%	2 min	5	0.1%
3	5	Success	98.9%	98.9%	1 min	9	0.2%

The accuracy (1) measures the degree to which the ontological information retrieved was useful for successfully implementing a change request. Accuracy is calculated as the as the number of changes that were implemented successfully (SC_i) by referring the Ontology (O) over the total number of change requests that arrived (TC). After a change is performed, it is necessary that the correctness (2) of the system including the relationship between the services is maintained. It also ensures that there are appropriate data flows (df) and control flow (cf) edges to the services that have been modified due to the change requests, it also assures that are no break points (bp), null (nr) or invalid references (ir) in the system. In our travel scenario discussed in section 4, after implementing the change request, we should ensure the data flow, for e.g. the information of flight details should be sent to hotel service. Similarly the control flow of the services, i.e where the control should be redirected after a service is executed. For e.g. in our scenario if a person has provided his details he should be redirected to the book flight service.

$$Accuracy = \sum_{i=1}^{n} \frac{\left(sc_i \cup \left(\frac{O'}{O}\right)\right)}{TC} \text{ --(1)}$$

$$correctness = [\sum_{i=1}^{n} \{\neg(bp + ir + nr) + df + cf\}] \times 100 \text{ ----(2)}$$

The time required for enacting a change also plays an important role in evaluating our framework; we aim to maintain this time minimum because the latest enterprises consider time as their major factor in their business. The change reaction time (3) for a system is calculated as the sum of time that is being spent in processing the change query (QP) and the time that spent for referring the ontology (OF) to achieve a successful change .Since ontology is tree based representation, the depth (L_n)till which the tree has been referred to implement a change is measured in terms of level of knowledge gained (4).

$$Change\ Reaction\ Time = \sum_{i=1}^{n} QP + OF \text{ - (3)}$$

$$Level\ of\ Knowledge\ gained = \sum_{SC_{i=1}}^{n} \{2 \times (L_n)\} - 1 \text{- (4)}$$

The Table.3 is calibrated when some of the requests fail to satisfy the analyst needs. This failure may arise as some of the services that are required to implement the change request may be unavailable. The MTBF (5) (Mean Time Between Failure) is difference the number of services that may be available ($Rser_i$) currently and the total number of services ($Rser_i$)that are required to perform the change to the number of requests that failed to implement (FC_i)..MTTR (Mean Time To Recovery) is the time that has been spent waiting for the unavailable services to become available for implementing the change. MTTR (6) (Mean Time To Recovery) is the time i.e. down time ($DTser_i$) that has been spent waiting for the unavailable services to become available for implementing the change over the number of requests that failed to implement (FC_i).

$$MTBF = \sum_{i=1}^{n} \frac{Aser_i - Rser_i}{FC_i} \text{--(5)}$$

$$MTTR = \sum_{i=1}^{n} \frac{DTser_i}{FC_i} \text{ -- (6)}$$

For all the cases, since our change management framework is based on automatic retrieval and evaluation, there may be some risk associated with this automation, which is measured with the help of the metric called Risk (7). From the highlighted data set in Table2, we find that by making changes using our change management framework has lower risk.

$$Risk = \sum_{i=1}^{n} 1 - \left\{\frac{SC_i \cap \left(\frac{O'}{O}\right)}{TC}\right\} \times 100 \text{ --(7)}$$

Table 3
Result Analysis for failure cases

No of change requests	No of change requests failed	LCS Size	Mean Time Between Failure	Mean Time To Recovery	Accuracy	Correctness	Level of Knowledge Gained	Risk
5	1	4	2	3	99.9%	99.9%	3	0.1%
3	1	5	2.1	3	99.9%	99.9%	3	0.1%
3	2	7	2.5	3.2	99.8	99.7	5	0.2%
2	1	5	0.9	2	99.9%	99.9%	7	0.1%

The Precision (8) and Recall (9) for our change management framework for both success and failure cases is plotted in Fig.10 and 11. It can be observed from the graph that the precision value for successful cases is found to be dominant over the recall values. On the other hand, the recall value is higher than the precision value during the failure cases because during the failure cases some of the relevant services become unavailable and hence the semantic information about the services could not returned by the system leading to an increase in the recall value.

$$Precision = \frac{g(x)}{g\prime(x)} \text{---(8)}$$

$$Recall = \frac{g(x)}{g(x)+f\prime(x)} \text{-- (9)}$$

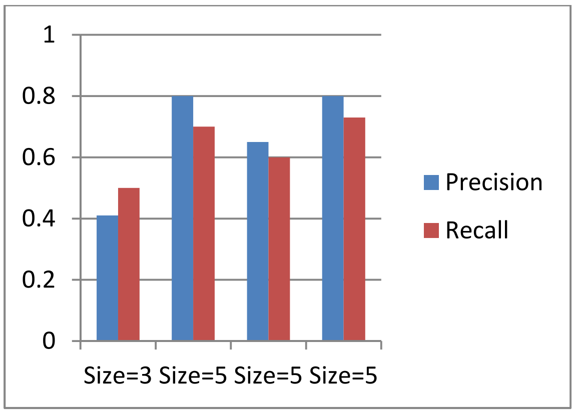


Fig10. Precision and Recall for Successful Cases

Precision is calculated as the number of functions that were found to be relevant ($g(x)$) over the total number of functions [containing both relevant and irrelevant ($g'(x)$)] returned by the system for making the changes. Similarly, recall is the number of functions that were found to be relevant ($g(x)$) over the summation of the total number of functions $g'(x)$ and the number of functions that are actually found to be relevant but not returned by the system, $f'(x)$.

7. DISCUSSION

7.1 Research Contribution

The Semantic Reasoner based change management framework contributes the following points and thus manages the evolving changes in the LCS. The research has the following salient features:

- Presents a parsing methodology to determine the possibility of implementing the requested change before it is being implemented at the service level with the help of the predictive parsing mechanism.
- Maintains the state information of the services involved in change enactment by the FSM methodology. This methodology is also responsible for automatic validation of the requested change.
- Presents a framework where the functionality of the services is also included in the ontology so that the more information about the services is provided to the analyst during the change enactment.
- Presents an environment for the analysts to implement the changes so that the cost and time spent on the IT professionals for making the changes is reduced.

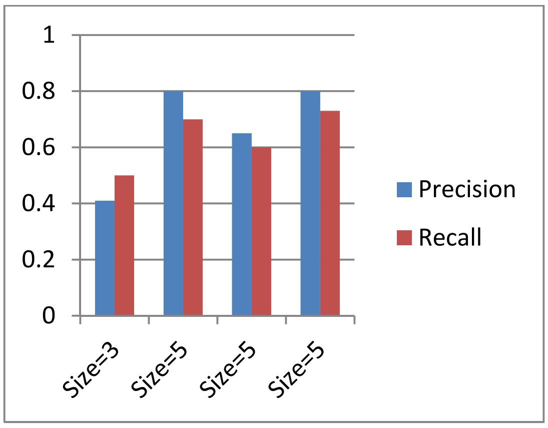


Fig11. Precision and Recall for Failure Cases

8. CONCLUSION AND FUTURE ENHANCEMENT

This paper proposes a Change Management Framework that satisfies the analyst requirements for implementing the change by referring an enriched ontology set. The creation of an enriched ontology leads to the reduction in the number of bugs, provides a better process clarity and also a clarification in the workflow .Enriching the ontology also reduces the percentage of risk thereby ensuring the accuracy of implementing the change. The parsing technique used in the framework ensures that the change request can be implemented before it is being implemented at the service level. The FSM that assists the analyst in the Change Management Framework also leads to the decrease in the overall computation time for implementing the change. Although, our framework determines the possibility of making the requested change previously so that the faults or disturbances that may arise during the change implementation at the service level are reduced, it is done only by the system. Therefore, our future goal can be to provide a sophisticated GUI at the schema level that will facilitate the analyst to analyse and implement the change. Finally, our framework can also be extended to support the implementation of concurrent change requests.

REFERENCES

[1]. Salman Akram,Xiaobing Wu,Virginia Tech, USA,Athman Bouguettaya,Xumin Liu,CSIRO ICT Centre, Australia,Armin Haller,Florian Rosenberg,Rochester Institute of Technology,USA," A Change Management Framework for Service Oriented Eterprises", *International Journal of Next-Generation Computing (IJNGC),* vol. 1, no. 1, pp 1-077, September, 2010.

[2]. Salman Akram,Athman Bouguettaya,Bramhim Medjahed,"Supporting Dynamic Changes to in Web Service Environments", *Springer-Verlag Berlin Heidelberg*, ISBN -978-3-540-20681-1, pp.319-334,2003.

[3]. Xumin Liu, Member, IEEE, Athman Bouguettaya, Fellow, IEEE, Jemma Wu, and Li Zhou," Ev-LCS: A System for the Evolution of Long-term Composed Services", *IEEE Transactions on Services Computing,* vol. 5, no. 2,pp.102-115, april-june 2010.

[4]. Xumin Liu , Athman Bouguettaya ," Managing Top-down Changes in Service-Oriented Enterprises", Web Services, 2007. ICWS 2007. *IEEE International Conference,*pp.1072-1079 July 2007.

[5]. Dimitris Apostolou, Gregoris Mentzas, Ljiljana Stojanovic, Barbara Thoenssen and Tomás Pariente Lobo, "A collaborative decision framework for managing changes in e-Government services", *Elsevier Journal of Government Information Quarterly*, vol. 28, no.

[6]. Aabhas V. Paliwal, Basit Shafiq,Jaideep Vaidya, Hui Xiong and Nabil Adam," Semantics-Based Automated Service Discovery", *IEEE Transactions on Services Computing*, vol. 5, no. 2,pp.260-275, april-june 2012.

[7]. Tamer A. Farrag, Ahmed I. Saleh, H.A. Ali," Semantic web services matchmaking: Semantic distance-based approach", *Elsevier Journal of Computers & Electrical Engineering*, Volume 39, Issue 2,pp.497-511, February 2013.

[8]. Meditskos, Georgios ; Dept. of Inf., Aristotle Univ. of Thessaloniki, Thessaloniki, Greece Bassiliades N, "Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S", *IEEE Transactions on Services Computing*, vol. 5, no. 2,pp.278-290, April-June 2010.

[9]. Carlo Mastroianni, Giuseppe Papuzzo," A self-organizing P2P framework for collective service discovery", Journal of Network and Computer Applications, Volume 39, pp: 428-437, March 2014, Pages 214-222.

[10]. Sourish Dasgupta, Anoop Aroor, Feichen Shen, And Yugyung Lee,IEEE," SMARTSPACE: Multiagent Based Distributed,Platform for Semantic Service Discovery", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 7,pp.805-821, July 2014

[11]. Jiujun Cheng, Cong Liu, mengchu Zhou*, Fellow, IEEE*, Qingtian Zeng, and Antti Ylä-Jääski," Automatic Composition of Semantic Web Services Based on Fuzzy Predicate Petri Nets", *IEEE Transactions on Automation Science and Engineering,* Volume:PP , Issue: 99 , ,pp. 1 – 10,28 January 2014

[12]. Son N. Han, Gyu Myoung Lee and Noel Crespi," Semantic Context-Aware Service Composition for Building Automation System", *IEEE Transactions on Services Computing*, vol. 5, no. 2,pp.752-761, April-June 2013.

[13]. Ourania Hatzi, Dimitris Vrakas, Mara Nikolaidou, Nick Bassiliades, Dimosthenis Anagnostopoulos, and Ioannis Vlahavas.," An Integrated Approach to Automated Semantic Web Service Composition through Planning", *IEEE Transactions on Services Computing*, vol. 5, no. 3,pp.319-332, July-September 2012

[14]. Szu-Yin Lin, Chin-Hui Lai, Chih-Heng Wu, Chi-Chun Lo," A trustworthy QoS based collaborative filtering approach for web service discovery", *Elsevier Journal of Systems and Software*, Volume 93, pp. 217-228, July 2014.

[15]. José Antonio Parejo, Sergio Segura, Pablo Fernandez, Antonio Ruiz-Cortés," QoS-aware web services composition using GRASP with Path Relinking",*Elsevier Journal of Expert Systems with Applications*, Volume 41, Issue 9, pp. 4211-4223,July 2014.