Framework for managing Business logic of web services through Schema generation and Property evaluation

Thirumaran. M
Lecturer, Dept. of CSE
Pondicherry Engineering College (PEC), India.

Dhavachelvan. P
Reader, Dept. of CSE,
Pondicherry University, India.

Asha.T
Student,
PEC, India

Lakshmi.P
Student,
PEC, India

ABSTRACT
Business Enterprise Management software needs to undergo structural modifications to gratify upcoming policies, whose advent may be due to business decision to satisfy customer demand or new business policy. Change management is a set of processes that is employed to ensure that significant changes are implemented to affect the organizational change. Here Change Management framework is proposed for making minor alterations to business logic but whose effect is more pronounced to the profitability of the organization. The main aspect is that commercial entity is managed at business analyst's discretion and not at developers' discretion which saves time and cost. A BLMF (Business Logic Management Framework) is a structured model in which a business analyst can store, retrieve, change and use the business rules that effect its operations in runtime itself. As business logic requirements change, business analysts can update the business logic without enlisting the aid of the IT staff. This business logic is made out at the run time so as to modify the logics in quick and better way. In this paper, the motivation for such framework by way of the genre of business products that the proposed architecture supports is presented. An account of proposed business logic management framework in terms of both functionality and the analyst friendly features available is detailed. Service Logic representation in XML schema exemplifies one of such analyst friendly features. All components are not product specification independent. Certain components of the framework are developed based on the product characteristics. Rule editor which helps identify the latent business rules in its logic is a product dependent component. Existing applications can be made compatible by developing an application compliant rule editor. Real time management is anticipated to get an edge over the existing management modules. So execution of this change management is discussed in service computing environment to throw light on how the service is modified in run time. Property Evaluation Engine is a noteworthy component of the framework. There has been a lot of research in computing and enhancing QoS parameters that aid fast retrieval of service but did not address fast modification of service and its impact analysis. Property evaluation engine is one such component that lends a hand in computing QoS parameters like Computability, Traceability, Time boundness etc. that help improve the reliability of change management system and guide change management process throughout its life cycle to increase its efficiency and robustness.

General Terms
Schema Generation, Property evaluation, Business logic and rules, Change management, Computability, Traceability

Keywords
Business logic schema, Business logic, Rule extraction, Source control management, Property evaluation, Web service maintenance, Impact analysis**.**

1. INTRODUCTION
Business rules ought to be changed off times to cope with the challenges in a free-enterprise economy. We focus on creating environment for dynamic variations to the business logic so that it is feasible to make frequent modifications in a service which aids service provider and serves him to satisfy clients' newfangled quests. Such environment is useful in the run-time management of web services and to exactly spot the solution to the service provider's maintenance element. There are many models that subsist for Business Process Management in which the process is recycled for the overall process changes. However, the problem with these solutions is that they only support the process level flexibility and not the application/service level flexibility. On the other hand this Business Logic Management framework tries to append that service level flexibility. These changes are done at the Business analyst level instead of being done at the Developer level, which reduces the hierarchy level in change management and thus implies a reduction in time and cost requirements. Here we propose a layered architecture for logic visualization and automated logic alteration which monitors the services at a point in time. The main goal is to bring forth a lucid Business Logic schema which versions the service source code according to the demands and requests raised by the user. The BL schema generation deals with generating a XML code that reflects changes to the service logic as anticipated and the rights to make modifications is handed over to authorized concerns wherein the security manager component comes in . We employ a request handler to process and filter the request and a source control manager for locating the services and a rule extractor which slices and segments the business rules. An execution planner is employed to handle the same request which arises over again. In the process we look forward to propose a property evaluator that appraises the dependability, computability, traceability, decidability and interoperability phenomena in business logic which brings about reliability to the components present. This paper draws a bead on providing immediate alterations to the services thus effectuating service automation process. Thus, it gives new dimensions to the

Business Logic community by discussing the transparent framework for Dynamic management of Business Logic.

2. RELATED WORKS

In this section, we discuss the academic research till now held in this domain. Business Process Management Systems in hand already, make the ongoing processes in business transparent to administrators so that they can tweak into and make changes whenever necessary. Claire Costello and Owen Molloy [1] in their paper introduced XESS (XML based Expert System Shell) that creates and downloads business rules to XESS Inference Engines deployed throughout a business scenario. Rule editing is accomplished with a rule editor and XESS Inference Engines bring in the new business rules into action. Here the kinds of rules that can be created are restricted in accordance with the rule editor capabilities. Business process oriented software architecture (BPOSA) for supporting business process change is being discussed by Qing Yao et al. Their paper also proposes the development of such architecture. [2]

The classical works of Harry M. Sneed & Katalin Erdos in regards with extraction of business rules present a tool named SOFTREDOC that not only extract business rules but also generates a data dictionary with the references to each data item, a procedure tree which depicts the structure of the component parts of a program, and a decision tree which represents the conditional logic of each program. Finally, it generates a table of program interfaces, both with other programs, i.e. CALL interfaces, and with the data environment i.e. data access interfaces, all of which aid in program maintenance.[3]

Michael zur Muehlen et al. presented the first contribution towards representational capabilities of process modeling languages and rule modeling languages and the conclusion was in favor of combination of these two i.e. BPMN and SRML together provided with a better representation of the business logic.[4]

And then regarding legacy systems, they are usually made of a lot of modules and business rule management gets tougher. Xie Gang in his paper proposes a approach for rule extraction which constitutes slicing legacy systems, domain variable identification, business rule extraction using dependence cache slicing and rule presentation and validation. Constructing dependence –cache slices requires static control dependence analysis and dynamic data dependence collection. [5]

Similarly overtime, business rules evolve and the software that implements it also gets morphed. As the encompassing software becomes large, the business rules embedded are substantial and difficult to extract. Chengliang Wang et al [6] have proposed a tailored solution approach to the rule extraction problem which consists of prime program slicing, prime domain variable identifying and data analysis, rule validation. Program slicing uses call graph approach to transform a large program into a smaller one that contains only statements relevant to the computation of a given function. Domain variable identification uses the heuristic rules for choosing domain variables of interest. This author [7] has already proposed a Business Logic model for web service source control management. There is also a detailed account on the cellular pattern generation for a particular web service that proves to be helpful in evaluating impact analysis once we manually generate the cellular pattern for modified web service.

3. ARCHITECTURE DIAGRAM

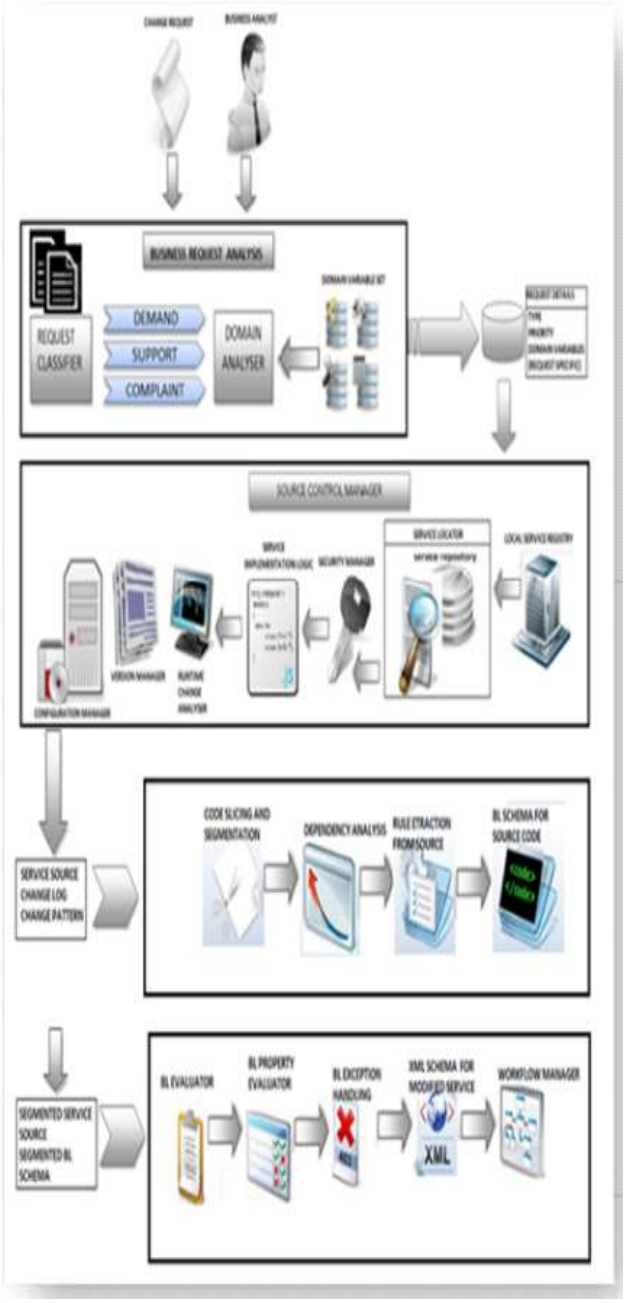

Fig 1.Architecture diagram

Business Analyst

A typical Business Analyst's technical understanding can be jotted down as follows. He possesses understanding in the areas of application programming, database and system design. He understands Internet, Intranet, Extranet and client/server architectures. He understands how legacy and web-based systems interface with each other. Here we authorize the analyst to customize the service infrastructure according to the requests raised in the dynamic business arena.

Change Request

A request that may be issued by end user or by employee working at workstation or by the analyst himself in need to adapt to a new business policy that demands in return a change to the currently available service or services. Request is a kind of abstract rule well formed according to specification of representation. In this section, the intricacies of architecture and its components are detailed. As mentioned in abstract, the course of action begins with issue of change request and the whole processing of the request is conducted under the supervision of an analyst. The framework detailed above constitutes four layers namely Business Request Analyzer, Source Control Manager, Business Rule Extractor and Business Logic Manager

Business Request Analyzer

This component is composed of two elements namely *Request Classifier* and *Domain Variable identifier*. Presented with a request as input, the Request Classifier classifies the given request into categories of Demand, Complaint and Support based on kind of request's representation, more specifically based on its specification. Priority is assigned in the order of Complaint, Support and Demand. This priority assignment assists analyst in making business decision of which request to be attended first. Domain Variable Identifier discovers the domain variables present in the request by just tokenizing the request and searching whether any of tokens match against given domain variable set. Once we are done with domain variable identification, we can determine the domain to which the request refers to i.e. we are also done with establishment of domain mapping. If the business product is magnificent, we employ tailored mechanisms presented by Xie Gang [base paper] for prime domain variable identification. This narrows the count of domain variables in case of large software systems.

Source Control Manager

We assert requests' domain to be the service domain and proceed with service discovery. With the present infrastructure available, we are able to manage get the service but not its source. Hence attainment of service's source pops in a need for security manager that takes the responsibility of authorizing concerned business people with the rights of access. The domain variables and domain specific to the request help search the local service registry and locate the service to be modified. Local service registry is privileged here to accommodate in it the source for every given service. Runtime change analyzer handles the issue of deciding if the change is feasible in runtime. If feasible the version manager creates a horizontal version i.e. a temporary copy which can be made permanent once all the expected changes are updated. Configuration manager facilitates implementation of a controlled change.

Business Rule Extraction

Business rules have got their hooks built deep into the business logic. Already a great deal of research work has been carried out in regards to this area of business rule extraction. In our component, the rules are extracted out and represented in XML kind of representation that augments its global appeal. Program slicing is applied to make the source more specific. First of all the service source is converted to schema for whose generation algorithm is given below. The way the code constructs are converted is shown in the template. The semantic structures are alone converted leaving out less meaningful syntaxes for variable declaration etc. From the XML kind of schema, it is easy to extract conditional usages, assignment usages for a given variable. This work is carried out by dependency analyzer. Both static and dynamic dependency analysis is employed. Static analysis discovers dependencies evident in the program structure say from conditional program constructs in a program, while a dynamic analyzer tests the scenario with an input, gets the workflow and searches for dependencies in entire path of information flow. Rules are nothing but a condition and its associated action. Rule set stores a service module and the rules extracted from it and if we wish to change the service, the corresponding rules' condition or action part is modified and the same is updated in services' source. In order to bring the change in action, it should be ensured that the change is made in service source and service redeployment and recompilation held so that end-user is available with the service.

Business Logic Manager

XML kind of schema generated is evaluated for its syntactic correctness by the Business logic evaluator and validated. Before the high-level language source alteration is done, a significant component of this architecture namely Business logic comes into picture. The aim of this component is to evaluate whether the properties of interoperability, traceability, decidability, computability are preserved even after the alteration. These properties are evaluated by business property evaluation engine and the calculated properties are appended to the schema to generate the final schema. The property values assist the analyst in making a decision whether or not to implement the alteration. Because of this reliability of the product is not much affected even after the change. Now the high-level language source alteration is done according to variations n schema. Service is redeployed and recompiled and ready for use.

4. WORKFLOW DESCRIPTION

In following sections, a detailed work flow diagram of the architecture is shown in Fig.2 The change request is made to follow certain norms for its representation for easy processing and the request classifier hence becomes business dependent. Once the representation pattern is defined, corresponding regular expression is formulated and the request is processed for the pattern to find to which category it belongs. Priority is assigned

to indicate the order of importance of the request. Requests' also seen in the same domain. Mechanisms presented by Xie Gang [5] for prime domain variable identification provides us with request specific domain variables that can be matched against Domain Variable Set to find the core domain. With the help of outputs from previous layers, WSDL files and Local service Registry that has in it service source and other access control details that aid in locating service source code. The source code is manipulated by schema generator to generate XML kind of representation which is understandable for analyst as well as machine. Business logic is monitored to find if it's modifiable in runtime and if the change made will stay reliable domain is the next area of interest because the effect of change is by specified components. Rules are extracted by methods suggested in [6] but using the generated schema as the foundation. Standard mechanisms are applied in for slicing and segmenting procedures. Conditional references and assignment usages are used for rule extraction in both static and dynamic dependency analysis. Rules have XML kind of representation, therefore easily processable. Rule set and service schema is evaluated for property determination and workflow management to finally obtain modified source followed by redeployment, recompilation and then service in action.

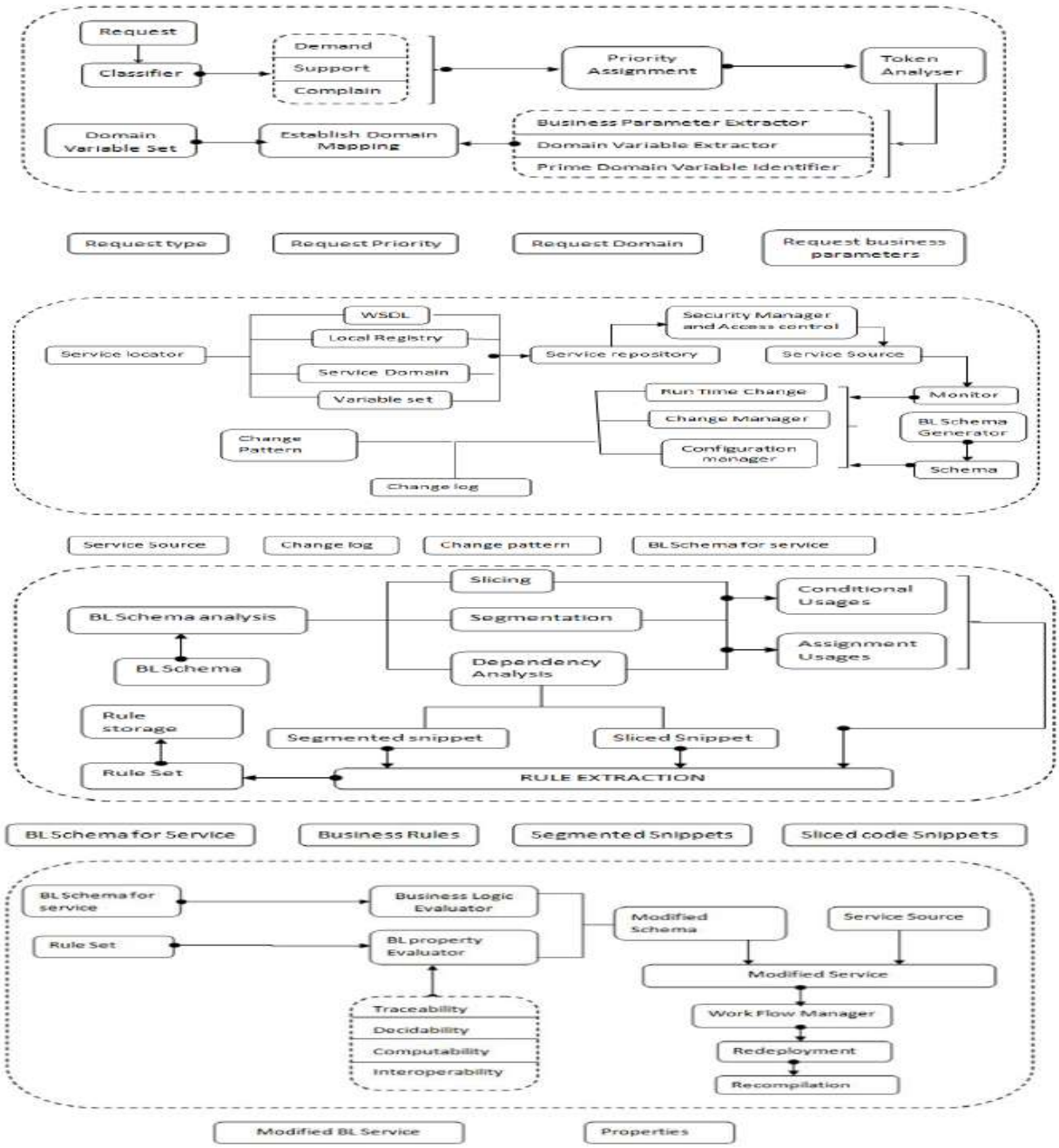

Fig.2 Model Diagram

5. 1 XML SCHEMA FOR C LANGUANGE LIKE CONSTRUCTS

The below table is used in the schema generator program for obtaining the template and for the program to gain knowledge of attributes that are to be accessed. The schema has got all possible sub elements and attributes that enforce the semantic structure of code even after the conversion to XML format.

SL.NO	STATEMENT	SAMPLE CODE	XML SCHEMA
1.	SELECT QUERY	select name,id from account where accno = +accno	<query type=”Data retrieval” name=”select” table=”account”> <rowset condition= “accno=+accno “> <columns> <column_name>name </column_name> <column_name>id </column_name> </columns> </rowset> </query>
2.	UPDATE QUERY	update account set curr_bal=curr_bal where accno=+accno	<query type =”DML” name=”update”> <rowset condition: “accno=+accno”> <columns> <column_name assignment=”curr_bal=+curr_bal”> curr_bal </column_name> </columns> </rowset> </query>
3.	INSERT QUERY	Insert into employee values (‘some_name’,’some_id’)	<query type=”DML” name=”insert”> <rowset > <columns> <column_name value=”some_name”>name</column_name> <column_name value=”some_id”>id</column_name> </columns> </rowset> </query>

4.	IF –ELSE STATEMENT	If(curr_bal<=amount)
		{
		Out.println(“Not enough money”);
		Return(“No”);
		}
		else
		{
		curr_bal=curr_bal_amt;
		return(“Yes”);
		}
			<if condition=”curr_bal<=amount”>
			<process>
			<ouput_statement>
			<print>“not enough money”</print>
			</output _statement>
			</process>
			<return> no </return>
			</if>
			<else>
			<process>
			<assignment_statement>
			<LHS> curr_bal </LHS>
			<RHS> curr_bal-amount </RHS>
			</assignment_statement>
			</process>
			<return> yes </return>
			</else>

5.	FOR LOOP	for(i=0;i<n;i++)
		{
		------ 	
		------ 	
		}
			<loop initial condition=”i=1” Condition=”i<n” Operation=”increment”>
			<process>
			---- 	
			</process>
			</loop>

6.	WHILE LOOP	while(i<=10)
		{
		---- 	
		---- 	
		}
			<loop condition =”i<=10”>
			<process>
			------ 	
			</process>
			</loop>

7.	SWITCH CASE	switch(choice)
		{
		case1:----------
		Break;
		case2:----------
		Break;
		default:----------
		Break;
		}
			<switch variable=”choice”>
			<cases>
			<case no=”1”variable_value=’A’>
			---- 	
			</case>
			<case no=”2”variable_value=’B’>
			---- 	
			</case>
			<default>
			---- 	
			</default>
			</cases>
			</switch>

8.	PRINTF	printf(“%d”,x); printf(“Welcome”);	<output_statement> <print>x</print> </output statement> <output_statement> <print>“Welcome”</print> </output statement>
9.	SCANF	scanf(“%d”,&x);	<input_statement> <get_input>x</get_input> </input_statement>

5.2 ALGORITHM FOR GENERATION OF XML SCHEMA FOR A GIVEN SOURCE CODE

As the name indicates, this is the procedure for conversion of program to XML schema.

GENSCHEMA (Source)
Begin
I:=0
LEVEL_INDEXER(Source,Level_table)
//to correlate structure begin with its associated end

While not end of source
Begin
Curr_token = READ_NEXT_TOKEN (Source)
For K:=0 to N /
/N is the size of structure having xml schema id,template,description etc.,

Begin
If(Curr_token = Table_keyword[K])
Begin
Lines[I] := Line_no
Line_id[I] := Table_id[K]
Template[I] := Table_template[K]
EXTRACT_PGM_SEGMENT(Source,Lines[I],Buffer,Level_table)
EndIf
EndFor
Curr_token :=READ_NEXT_TOKEN(Source)
EndWhile
End
//----------

GET_ATTRIBUTE(Buffer,Line_Id,Attribute)
Begin
While not end of Buffer
Switch(Id)
Begin
Case1: GET_ATTRIBUTE_LOOP(Buffer,Attribute,Line_Id)
Case 2:
GET_ATTRIBUTE_SELECT_QUERY(Buffer,Attribute,Line_Id)
Case 3:
GET_ATTRIBUTE_CONDITIONAL_CONTROL_STRUCTURE(Buffer, Attribute,Line_Id)
.....
.....//cases covering all program structures;
EndSwitch
End
//----------

```
GET_ATTRIBUTE_LOOP(Buffer,Attribute,Line_Id)
//procedure for extracting attribute values for loop Begin

Attrib_value :=
EXTRACT_INITIAL_CONDITION_VALUE(Buffer)
Write to Atribute file
Initial_condition := Attrib_value
Attrib_value := EXTRACT_CONDITION_VALUE(Buffer)
Write to Atribute file
Condition := Attrib_value


Attrib_value := EXTRACT_OPERATION_VALUE(Buffer)
Write to Atribute file
Operation := Attrib_value
End
//---------------------------------------------------------------------

SUBSTITUTE_ATTRIBUTE(Attribute,Template,Partial_out
put)
Begin
        While not end of template
        Begin
        Curr_token := READ_NEXT_TOKEN(Template)
        Find    :=    CHECK(Empty_attrib_value_field_to
        be_filled)
        If(Find)
                Begin
                        Prev_token :=
        READ_PREVIOUS_TOKEN(curr_token,Template)
        While not end of  Attribute
        Begin
        If(Prev_token = Attribute_name)
        Begin
        Value :=
        GET_ATTRIB_VALUE(Attribute,prev_token)
        Write to Partial_output file
        Curr_token := READ_NEXT_TOKEN(Template)
         EndIf
EndWhile

Else
        Write to Partial_output file
        Curr_token := READ_NEXT_TOKEN(Template)
EndIf
EndWhile
//---------------------------------------------------------------------

LEVEL_INDEXER(Source,Level_table)
While not end of Source file
        Begin
        I :=0
       Curr_token=READ_NEXT_TOKEN(Source)
       If(Curr_token = Pgm_structure_begin)
Begin
 Starttag_lineno[I] :=  Line_no
Curr_token=READ_NEXT_TOKEN(Source)
Increment I
ElseIf(Curr_token=pgm_structure_end)
Decrement I
Endtag_lineno[I] :=Line_no
Curr_token=READ_NEXT_TOKEN(Source)
Increment I

Else
Curr_token=READ_NEXT_TOKEN(Source)

EndIf
EndWhile
        Level[0] := 0
        Level[1] := 1
For  K := 2 to I-1
        Begin
        If(Endtag_lineno[K]<Starttag_lineno[K-1])
        Begin
        Level[K] := Level[K-1]+1
        Else
        Level[K] := Level[K-1]
EndIf
EndFor
For  K := 0 to I-1
        Begin
        Open Source file
        GOTO(starttag_lineno[K])
        If(Level[K]=0)
                Begin
        Append parent tag
        GOTO(endtag_lineno[K])
                Append closing parent tag
        ElseIf((Level[K]=1)
                Append child tag
                GOTO(endtag_lineno[K])
                Append closing child tag
        ElseIf((Level[K]=2)
                Append grandchild tag
                GOTO(endtag_lineno[K])
                Append closing grandchild tag
        Else
        Print"Only three levels of nesting supported"
        End program execution
EndIf
EnfFor
End
//---------------------------------------------------------------------

EXTRACT_PGM_SEGMENT(Source,Lines[I],Buffer,Level_
table)
Begin
While not end of Level_table
Begin
If(Lines[I] != Starttag_lineno[K])
Begin
While not end_of_statement
        Begin
        Write to Buffer File
EndWhile
GET_ATTRIBUTE(Buffer,Line_id[I],Attribute)
```

```
SUBSTITUTE_ATTRIBUTE(Attribute,Template[I],Buffer,Part
ial_output)
Append Partial_output content to Output
        Else
        If(Starttag_lineno[K+1]> Endtag_lineno[K])
        Begin
        For Line_no=Starttag_lineno[K] to Endtag_lineno[K]
        Begin
        Write to Buffer file
        EndFor
GET_ATTRIBUTE(Buffer,Line_id[I],Attribute)
SUBSTITUTE_ATTRIBUTE(Attribute,Template[I],
        Buffer,Partial_output)
        Append Partial_output content to Output
EndIf
        If(Endtag_lineno[K+1]<Endtag_lineno[K])
                Begin
        EXTRACT_PGM_SEGMENT(Source,
        Starttag_lineno[K+1],Buffer,Level_table)
        EndIf
EndIf
EndWhile
End
//---------------------------------------------------------------------
```

The source code is given as input to the program. The program processes the code to give desired schema. The code is level indexed first i.e. blocks, inner blocks and inner most blocks are identified in case of presence of nested if..else and nested for loops. This is for aiding highly understandable schema. Then each line is scanned. Within each line the tokens are searched for their entry in the table shown before. If found, the corresponding schema template is retrieved for future use and the code part that should be converted is extracted. Extract_pgm_segment () method is employed for this. It takes into consideration all possible kinds of program constructs and extracts code segment from beginning to ending. Getattributes () method extracts appropriate attributes depending upon the kind of program construct. If the select DB query is to be processed, then table name, column names, row set condition etc.attributes are extracted. Setattributes substitutes the attributes into the template. The entire code is processed the same way and merged to get the schema.

5.3 EXAMPLE

Here is an example of a withdrawal procedure in a bank application being converted into XML schema which is accomplished by the above algorithm. The possible rules are extracted from the XML schema. According to the change request the rule sets are modified in the XML schema.

Source program for withdrawal

```
Public string withdrawal ( int acc_no, double amount)
{
sql = "select * from account where acc_no = "+accno""
rs.executeupdate ( );
amount = 500;
if (curr_balance <= amount)
{
        out. print ("Balance is not enough");
        return ("No");
}
else
{
        curr_balance=curr_balance - amount;
        return("yes");
}
sql = "update account set curr_balance = "+curr_balance"
where acc_no="+acc_no""
rs.executeupdate ( );
}
```

Modified rule set as per the request

```
//rule #1
<assignment_statement>
        <LHS>amount</LHS>
        <RHS>600 </RHS>
</assignment_statement>
//assignment usage of amount variable
//rule #2
// No change
```

Modified schema

```
<assignment_statement>
        <LHS>amount</LHS>
        <RHS>600 </RHS>
</assignment_statement>
```

Modified source

```
amount=600;
```

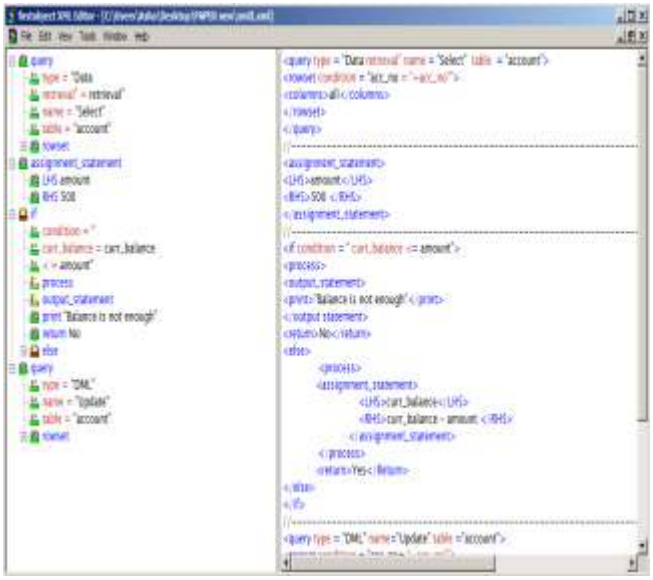

Fig 3.Schema for the above source program for bank withdrawal

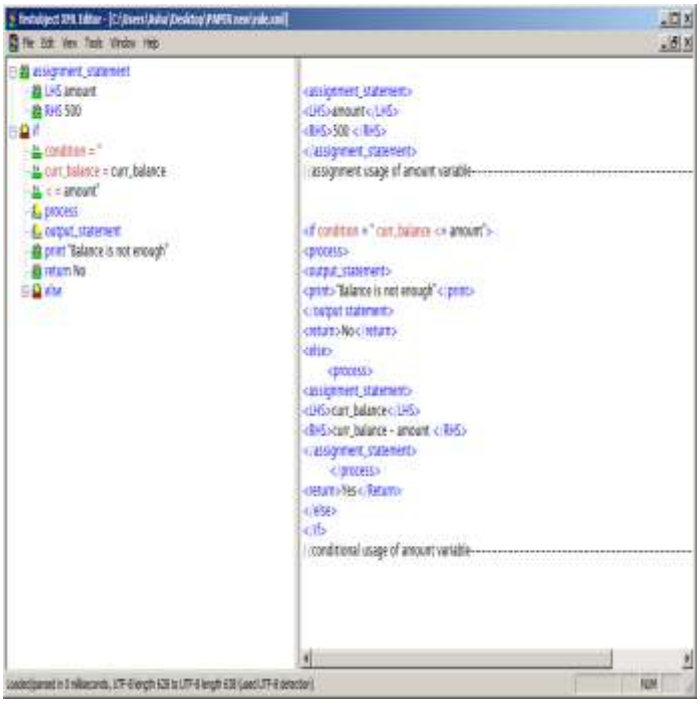

Fig .4 Rule set extracted from above source

6. PROPERTY EVALUATION

Computability

Computability refers to feasibility of the modification anticipated. Whenever the request is issued, it is being edited by the analyst as a rule in order to bring in action the change request using the rule editor whose snapshot is presented in figure below. This editor allows only certain type of requests to be made i.e. computability of the request is determined by this editor. If the request can be edited as rule using this editor, it means it is computable. And hence computability of a request is determined to be true or false based on whether it can be formed as rule by the editor or not. Therefore rule editor should consider all intricacies of the business product in its construct to effectively able to represent all computable rules. Thus the functional QoS parameter, computability is identified. It is enhanced by the construction of a functionally complete rule editor.

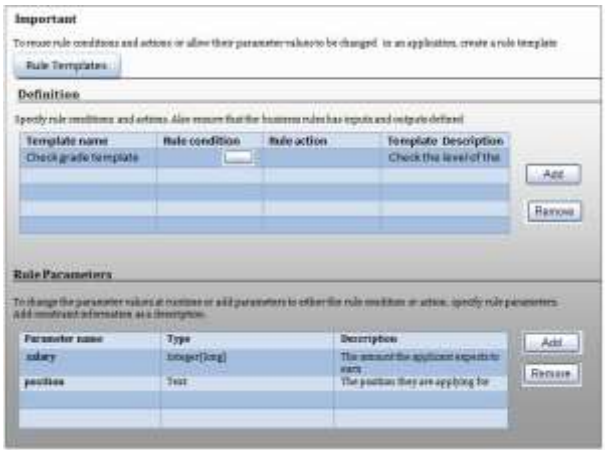

Fig 5 Rule Editor

There are also other kinds of computability's defined, partial and complete computability. Say a rule includes a set of actions. If all the requirements of the action are satisfied it is said to be completely computable. If few of the requirements of the action are satisfied it is said to be partially computable. Say an action is composed of a data retrieval statement. If the specified database, table, fields are available, it is said to be computable. So in the editor we have also the space to specify actions constituted by a rule and we can calculate computability to provide more insight into computability of the rule.

PROPERTY IDENTIFIED: Computability of a request
VALUE: True/False
ESIMATION PHASE: Business Request analysis phase
ENHANCED BY: Functionally complete Rule editor

Rule Traceability

Rule traceability refers to tracing of similar rules which were solved before and issue of using its execution plan in solving current issue. Some errors may occur during runtime. The complaint is given by the user in general English and tries to edit it as rule using editor.

In case of the same error occurring at two different locations, he may give the complaint in different way. If we manage to store the thread control block of that session (shown below),

then it is easy to identify similar change requests whose origin is due to the same internal operating state of the service. If the request happens to match and the issue was solved, we flag the present request as repeating request and indicate that the issue was not solved completely. If it happens to match partially, say parameters alone, we can make use of execution plan of the change request solved before.
PROPERTY IDENTIFIED: Rule traceability of a request
VALUE: (True, Execution plan)/(False, Repeating request)
ESIMATION PHASE: Business Request analysis phase
ENHANCED BY: Execution planner

7. CONCLUSION

The proposed framework is a full fledged one, when all the components of it are completely conceptualized will turn out to be a great tool for business change management i.e. runtime time change management. In this paper, we focused on defining the components, where few components had already its base derived from the research work available. Here we define the need for components such as property evaluator and run time manager. The novel procedure introduced here is the XML schema generation even for the programming logic statements and making that as base for rule extraction sample case study where a withdrawal module of banking program is managed making use of the above discussed concepts depicts the simplicity and effectiveness of the architecture.

REFERENCES

[1] Claire Costello and Owen Molloy, "Orchestrating Supply Chain Interactions using Emerging Process Description Languages and Business Rules", Sixth International Conference on Electronic Commerce,ICEC,pp 21-30,2004.

[2] Qing Yao,Jing Zhang,Haiyang Wang, "Business Process Oriented Architecture for Supporting Business Process Change", International Symposium on Electronic Commerce and Security,pp 690- 694,2008.

[3] Harry M. Sneed , Katalin Erdos, "Extracting Business Rules from Source Code", IEEE publication,2004.

[4] Michael zur Muehlen, Marta Indulska, Gerrit Kamp, "Business Process and Business Rule Modeling Languages for Compliance Management: A Representational Analysis", Twenty-Sixth International Conference on Conceptual Modeling, 2007.

[5] Xie Gang, "Business Rule Extraction from Legacy system Using Dependence-Cache Slicing", The 1st International Conference on Information Science and Engineering, ICISE,pp 4214-4218,2009.

[6] Chengliang Wang , Yaxin Zhou, Juanjuan Chen, "Extracting Prime Business Rules from large legacy system", International Conference on Computer Science and Software Engineering, pp 19-23, 2008.

[7] Thirumaran.M, Dhavachelvan. P, Tushar Ranjan Sahoo, Maria Stephen "Business Logic Model for Web Service Source Control Management" International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 21, 2010

[8] Bassam Atieh Rajabi, Sai Peck Lee, "Change Management in Business Process Modeling Survey", *2009 International Conference on Information Management and Engineering.*

[9]. F. A. Blaauboer, K. Sikkel, M.N Aydin, "Deciding to Adopt Requirements Traceability in Practice", In *Proc. of 19th Int. Conf. on Advanced Information Systems Engineering (CAiSE'07)*, Springer Lecture notes in Computer Science 4495, Norway, 2007, pp. 294-308.

[10]. J. Cleland-Huang, "Toward Improved Traceability of Non-Functional requirements", *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, Long Beach, California, 2005, pp. 14 – 19.

[11]. Uttam Kumar Tripathi, Knut Hinkelmann, "Change Management in Semantic Business Processes Modeling", *Eight International Symposium on Autonomous Decentralized Systems (ISADS'07).*

[12]. Xiang Luo, Koushik Kar, Sambit Sahu, Prashant Pradhan, Anees Shaikh, "On Improving Change Management Process for Enterprise IT Services" *IEEE International Conference on Services Computing, 2008.*

[13] Alpigini, J.J., Neill, C.J., and Ramsey F.V., "Classification of Rule Extraction Techniques from Knowledge-Based Systems", *Proceeding of the IASTED International Conference Modeling and Simulation*, 2001, pp. 60-64.

[14] X. Wang, J. Sun, X. Yang, Z. He, and S.R. Maddineni, "Human Factors in Extracting Business Rules from Legacy Systems," in Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics, 2004, pp. 200-205.

[15] Software AG, "Unlock data and functions in your legacy systems", GrossVision, 2006.

[16] O.Vasilecas and A. Smaizys, "Business rule specification an transformation for rule based data analysis," in Proc. of 19th International Conference on Systems for Automation of Engineering and Research (SAER 2005), R. Romansky, Ed., 2005, pp. 35-40.

[17] M. Bajec and M. Krisper "A methodology and tool support for managing business rules in organizations", Information Systems, 2004, Elsevier

[18]. M. Bajec and M. Krisper "A methodology and tool support for managing business rules in organizations", Information Systems, 2004, Elsevier.