A Navel Approach for Evaluating Completeness of Business Rules using First Order Logic

M.Thirumaran, E. Ilavarasan, R. Manoharan, Thanigaivel.K

Abstract - Organizations are under increasing scrutiny to develop the rules to regulate their business requirements and it is important to standardize the business rules to automate the complex process into simple logic. So several process modeling languages and rule modeling languages are evolved to modulate the organizational policies and procedures. Business rules are the constraints which influence the behaviors and also specify the derivation of conditions that affect the execution flow. These rules are form of conditional operations attached to the process to give data result. These systems implement business rules that are restructured when organizations change the data to the varying business needs. When these rules are changed, it must be efficient to provide a decision based on the given constraints or based on business requirements. The correct decision logic are verified by evaluating or validating the completeness of the business rule. More often the rules can associate with another rule to derive business logic, but these rules are not complete enough to determine the computability of business logic. So it is necessary to check whether the rule is complete, this can be proved when the rules are interpreted with first order logic. This paper provides standard approach to evaluate the completeness of business rules by formulating the rules using first order logic. Applying these strategies introduces a structured approach and management aspects within rules by focusing on rule sources which are important for the process goal and providing a meaningful rule structure. It points out the necessity and also the real possibilities to establish new facilities for manipulation of business rules into the software development process. This can give rise increase in the performance of the legacy system.

Index Terms - Completeness, Rule Execution, Business process, Business rules, Business Rules Approaches, First Order Logic, Business rules, First Order logic, Completeness algorithm.

1. INTRODUCTION

Many efforts are already made to promote and integrate business rules modeling and management tools into business processes automation. Being ready for execution as a part of business process management applications business rules and the business rules approach still seem to lack a common methodological ground and support. Especially the fact that, before business rules can be modeled and managed they first need to be captured and extracted from different sources is often left to analyst's intuition. Well- known possible sources to look at to identify business rules are, among others: process documentation, source code or implicit sources like internal problem-solving knowledge of the employees involved in this process. But since not many enterprises capture their business rules in a structured, explicit form like documents or implicit software codes, they need to be identified first, before being captured and managed. Here the question about how to identify the potential sources of business rules emerges. Though some of them may seem obvious, like legal restrictions, standards or mandatory best practices, some are implicitly included within the elements involved in the process. On the other hand, many enterprises capture their business processes in business process models, providing a structured view for further analysis and management. Business is performed according to rules. They make an important and integral part of each information system (IS) by expressing business logic, constraints on concepts, their interpretation and relationships. Therefore is relevant to pay special attention to business rules in development of information systems. Defined rules of structure and behavior are captured in the models of structures, states, processes and other IS models. In general, models can be expressed using graphic symbols, text and formulas. Modeling constructs are grouped into a variety of diagram types by various aspects, for example, process, static structure, states. These rules are need to check for consistency and completeness. In order to facilitate the completeness of the rules we are using First Order Logic.

2. LITERATURE SURVEY

Business Process Management (BPM) is an established discipline for building, maintaining, and evolving large enterprise systems on the basis of business process models. A business process model is a flow-oriented representation of a set of work practices aimed at achieving a goal, such as processing a customer request or complaint, satisfying a regulatory requirement, etc. The Business Process Modeling Notation (BPMN) is gaining adoption as a standard notation for capturing business processes. The main purpose of business process models generally, and BPMN models in particular, is to facilitate communication between domain analysts and to support decision-making based on techniques such as cost analysis, scenario analysis, and simulation Chun Ouyang [1] proposed an acyclic BPMN model, or an acyclic fragment of a BPMN model, falls under the class if it satisfies a number of semantic conditions such as absence of deadlock. He applied Petri net analysis techniques to statically check these semantic conditions on the source BPMN model. According to Michael zur Muehlen [2], Business Processes are sets of activities that create value for a customer. While research in Business Process Management initially focused on the documentation and organizational governance of processes, organizations are increasingly automating processes using workflow systems, and are building elaborate management systems around their processes. Such management infrastructures integrate modeling, automation, and business intelligence applications. The inclusion of compliance management activities is a logical next step in governing the business process life cycle. Josef Schiefer [3] says that Business Process Management (BPM) systems are software solutions that support the management of the life cycle of a business process. For the execution of business processes, many organizations are increasingly using process engines supporting standard-based process models (such as WSBPEL) to improve the efficiency of their processes and keep the testing independent from specific middleware. A major challenge of current BPM solutions is to continuously monitor ongoing activities in a business environment and to respond to business events with minimal latency. One of the most promising concepts that approaches the problems of closed-loop decision making and the lack of gaining real-time business knowledge is the concept of Complex Event Processing (CEP), The system automatically discovers and analyzes business situations or exceptions and can create reactive and proactive responses, such as generating early warnings, preventing damage, loss or excessive cost, exploiting time-critical business opportunities, or adapting business systems with minimal latency. Claire Costello [4] says A business process as a complete set of end-to-end activities that together create value for the customer. Business process management is the ability to orchestrate and control the execution of a business process across heterogeneous systems and allow users to view the components of an infrastructure from a process view rather than set of applications and databases. Anca Andreescu [5] says every organization operates according to a set o business rules. These may be external rules, coming from legal regulations that must be observed by all organizations acting in a certain field, or internal rules which define the organization’s business politics and aim to ensure competitive advantages in the market. Starting from the previous observations, it is obvious the important role that business rules play within the development process of a software system. Milan Milanović1 [6] proposed that BPM languages have limited support for representing logical expressions, business vocabularies, and business rules, which severely limits their flexibility and expressivity. To address these challenges, he integrated business rule modeling constructs of the REWERSE Rule Markup Language (R2ML) with the Business Process Modeling Notation (BPMN), resulting in a *rBPMN* proposal. Olegas Vasilecas [7] says Business rules make an important and integral part of each information system (IS) by expressing business logic, constraints of concepts, and their interpretation and relationships. Therefore it is relevant to pay special attention to business rules in development of information systems. Rules related to domain structure and behavior are presented in data, states, processes and other IS models. Taking into account that rules are expressed in several models, there is a risk that overall specification is inconsistent. Unambiguous models are crucial for the successful implementation of IS models transformation and finally code generation tasks. Therefore it is necessary to check consistency among related rules models. The problem of models inconsistency can be solved by using of formal or partially formal models with constraints. However formal models are often too complex to be used in practice. Semi-formal models are widely used, but constraints used in such a models often are suitable only for one model and relationships among models are not defined. He suggests extending of IS approach based on semi-formal models and constraints, by adding the consistency rules for IS models. Anis Charfi [8] applied the divide and conquer principle to web service composition by explicitly separating business rules from the process specification. The combination of the business rules approach with the process-oriented composition solves a twofold problem. First, he provided a solution to the problem of dynamic adaptation of the composition. In fact, current standards for process-based web service composition are not capable to deal with the flexibility requirements of composite web services. Second, business rules are important assets of a business organization that embody valuable domain knowledge. So, it is no longer acceptable to bury them in the rest of the composition. Bruno de Moura Araujo [9] proposed Business rules (BR) are declarations which constrain, derive and give conditions for existence, representing the knowledge of the business. BRs are not descriptions of a process or processing. Rather, they define the conditions under which a process is carried out or the new conditions that will exist after a process has been completed. BR can be represented using Semantics of Business Vocabulary and Rules (SBVR). SBVR is appropriate to be used by business experts, since it allows the representation of business vocabulary and rules using controlled natural language. Business vocabulary concepts can be automatically transformed into conceptual models, like the UML class model. Nicholas Zsifkov [10] says Enterprise business rules are usually defined as constraints or as metadata about business operations: on the business side, business rules are special policies that define constraints/metadata about the business operation; on the information system (implementation) side, business rules are constraints about the data, about data manipulation and about system processes. Antonio Oliveira Filho [11] proposed a new traceability technique that defines dependency links with the same semantics that can be observed in the relationships among business rules. The goal of the technique is to provide rates of recall and precision of 100% for changes in software requirements that correspond to business rules. Jose F. Mejia Bernal [12] says decomposing the initial business process structure in a set of rules is a procedure based on pattern identification. This approach consists of two phases: mapping of business process to rules, and reliable adaptation of the business process according to the context data. The first phase is executed to provide a representation of the initial business process definition in terms of rules. The second phase is applied to provide a reliable workflow process modification.

Timon C. Du and Hsing-Ling Chen propose an active collaboration and negotiation framework (ACNF), which is a negotiation support system that uses active documents with embedded business logics or business rules that can adapt to different collaborative strategies in a business-to-business (B2B) environment [13]. Sam Weber and Isabelle Rouvellou describe a fully functional prototype middleware system which provides the users to control software components without the need of programming knowledge. Thus the core applications need not be altered for anticipated changes from external factors. In this system, application behavior modification is fast and easy, making this middleware suitable for frequently changing programs [14]. H. M. Sneed[15], in the context of reverse engineering source code into UML diagrams many tools and approaches have been developed.CPP2XMI is a reverse engineering tool which lows extracting UML class, sequence and activity diagrams in XMI format from C++ source code. Sangseung Kang[16], Business rules are business statements that define some aspect of a business. They describe, constrain and control the structure, operations and strategy of the business. In his paper, he analyzes business rules and business rule systems, and present requirements and considerations for the business rule expression and system. Olga Levina[17], suggests an extraction process for business rules identification from business process models. Applying this process introduces a structured approach and management aspects within rules discovery by focusing on rule sources that are important for the process goal and providing a rule structure. Mohammed Alawairdhi[18], suggests a business-logic-based framework for evolving software systems is proposed. The goal of the framework is evolving software in a higher abstract layer. Olegas Vasilecas[19], suggests The rules are expressed in several models, there is a risk that overall specification is inconsistent. Unambiguous models are crucial for the successful implementation of IS models transformation and finally code generation tasks. Therefore it is necessary to check consistency among related rules models. The problem of models inconsistency can be solved by using of formal or partially formal models with constraints. According to Anis Charfi[20], the Business Rules Group , a business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control the behavior of the business. Business rules are usually expressed either as constraints or in the form if conditions then action. The conditions are also called rule premises. The business rule approach encompasses a collection of terms (definitions), facts (connection between terms) and rules (computation, constraints and conditional logic). Terms and Facts are statements that contain sensible business relevant observations, whereas rules are statements used to discover new information or guide decision making. Jose F. Mejia Bernal[21], proposed a way for representing Dynamic Business Process in terms of Rules based on patterns identification. With this approach it is easy to apply on a business process instance both user-based personalization rules and automatic rules inferred by an underlying context-aware system. Gulnoza Ziyaeva proposed framework to enable the content-based intelligent routing path construction and message routing in ESB which defines the routing tables and mechanisms of message routings and facilitate the service selection based on message content [22].

3. Rules for Car license

	Rule	Syntax	JESS Syntax	Jrule	First Order Logic
1	Car must not be rented to customers without a valid license number.	if(Customer.ValidLicenceNumber == "FALSE") { application. Status = "Reject"; Customer.El	(def rule no car rented without lenience no) (if(=(Customer.ValidLicenceNumber ? true) =>(add(application.status)(/?reject) (add(customer.eligible)(/?false)	(def rule no car rented without lenience no) when (Customer.ValidLicence Number =? true) => then (application.status)(?reject)	∃ *y[customer id(y)]* ∧ *z[licence no(z)]* $\nsim$ $f(false(f)$ → *application status(reject(y,z)).*

		igibile = false; }		(customer.eligible)(?false)	
2	Car must not be rented to customers of Age less than 18.	if(Customer.age < 18) { application.Status = "Reject"; Customer.Eligibile = false; }	(def rule no car rented to customer of age less than 18) (if(=<(customer.age ? 18)(age?age) =>(add(application.status)(/?reject) (add(customer.eligible)(/?false)	(def rule no car rented to customer of age less than 18) when (customer.age ? =< 18)(?age) => then (application.status)(?reject) => (customer.eligible)(/?false)	∃ *y[customer.age(y)]* ≤ 18 → *application status[valid(y)]* ∧ *c [customer eligible(c,y)]*
3	Car must not be rented to customers with bad history level 3	if(Customer.BadHistoryLevel == 3) { application.Status = "Reject"; Customer.Eligibile = false; }	(def rule no car to customer of bad history level 3) (if(=(Custome.badhistorylevel ? 3) =>(add(application.status)(/?reject) (add(customer.eligible)(/?false)	(def rule no car to customer of bad history level 3) when (Custome.badhistorylevel= ? 3) => then (application.status)(?reject) (customer.eligible)(?false)	∃ *y[customer.bad historylevel(y)]* ≤ 3 → *application status[reject(y)]* ∧ *c [customer eligible(false (c,y)]*
4	Rent for small cars is 80 aud per day	if (Customer.Eligible == true) { if(Car.type == Small) { ent.RentPerDay = 80; }	(def rule rent for small car is 80/day) (if(=(Customer.Eligible ? true) { if(=(cartype?small) { =>add(rent.perday)(/?price 80)) }	(def rule rent for small car is 80/day) when (Customer.Eligible=? true) { if(cartype=?small) { then => (rent.perday)(?price 80)) }	∋*y[customer.eligible=true(y*) ∧ $c(car\ type = small(c)$ *] → r[(rent per day(80,y,c)]*
5	Rent for awd cars is	if(Car.type == AWD)	(def rule rent for awd cars 100/day)	(def rule rent for awd cars 100/day)	∋*y[car.type=true(y,AWD(*

	100 aud per day.	{
		rent.RentPerDay = 100;
		}
			(if(=(car.type?AWD)
			{
			=>add(rent.perday)(/?price 100)
			}
				when
				(car.type=?AWD)
				{
				=>
				then
				(rent.perday)(?price 100)
				}
					c))] → r[(rent per day(100,y,c)]

6	Rent for luxury cars is 150 aud per day	if(Car.type == Luxury)
		{
		rent.RentPerDay = 150;
		}
			(def rule rent for luxury cars 150/day)
			
			(if(=(car.type?luxury)
			{
			=>add(rent.perday)(/?price 150)
			}
				(def rule rent for luxury cars 150/day)
				when
				(car.type=?luxury)
				{
				=>
				then
				(rent.perday)(?price 150)
				}
					∃ y[car.type=true(y,luxury car(c))] → r[(rent per day(150,y,c)]

7	Rent payable is calculated as the product of rentperday and rentalperiod in days.	{
		rent.RentPayable = rent.RentPerDay * rent.No of rent days;
		if (CustomerBadHistoryLevel > 0)
		}
			(def rule rent calculated as the product of rentperday and rentalperiod in days)
			
			(rent.rentpayable)(=(*(?rentperday?rentalperiod)))
			(if(>(Custome.badhistorylevel ? 0)
			=>(add(rent.rentpayable)(price?price)
				(def rule rent calculated as the product of rentperday and rentalperiod in days)
				when
				(rent.rentpayable)*(?rentperday=?rentalperiod)))
				(if(>(Custome.badhistorylevel ? 0)
				=>
				then
				(rent.rentpayable)(price ?price)
					∃ y[rent.payable]=rent.per day(z)]*no of rent per day(a)] → c(customer .bad history level>0) → (z,a,c)

8	Penalty of 20 % of rent must be applied for customers with bad history level 2.	if(Customer.BadHistoryLevel == 2
		{
		rent.PenaltyFee = rent.RentPayable * 0.2;
		}
			(def rule Penalty of 20 % of rent for customers with bad history level 2)
			
			(if(=?Customer.BadHistoryLevel?2)
			{
			=>
			add(rent.penaltyfee)(=(*(?rent.rentpayable?0.2)))
				(def rule Penalty of 20 % of rent for customers with bad history level 2)
				when
				(?Customer.BadHistoryLevel=?2)
				{
				=>
				then
				(rent.penaltyfee)*(?rent.rentpayable?0.2)))
					∋ y[customer.badhistorylevel(y)]=p[Penality.fee(p)] → r[(rent.payable(r)*0.2)]

9	Penalty of 10 % of rent must be applied for customers with bad history level 1.	if(Customer.BadHistoryLevel == 1) { rent.PenaltyFee = rent.RentPayable * 0.1; } }	(def rule Penalty of 10 % of rent for customers with bad history level 1) (if(=?Customer.BadHistoryLevel?1) { => add(rent.penaltyfee)(=(*(?rent.rentpayable?0.1)))	(def rule Penalty of 10 % of rent for customers with bad history level 1) (if?Customer.BadHistoryLevel=?1) { => then (rent.penaltyfee)*(=(?rent.rentpayable?0.1))	∋ *y[customer.badhistorylevel(y)]=p[Penality.fee(p)*] →*r[(rent.payable(r)*0.1)]*

4. 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 choice 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 extra complications come about because of these restrictions, needed to guarantee soundness of the rules , on the status of the variables. An algorithm has been proposed by considering the above standards.

$$L=\{R,F,P,D\}$$

Let 'L' be the Business Logic, which consists of 4-tuples such as R – Set of Rules, F- Set of Functions,
P- Set of Parameters
D- Set of dependency relation

Completeness Theorem

L={R,F,P,D}
Let 'L' be the Business Logic, which consists of 4-tuples such as
R – Set of Rules
F- Set of Functions
P- Set of Parameters
D- Set of dependency relation

Theorem: *The Business Logic 'L' is complete iff the associated Rules, Functions, Parameters and the dependency relations are complete.*

Proof:
Parameters are complete

Input Parameters <I_1,I_2,I_3> are bound with discrete values then return Boolean value (True)

$$B(p) = \begin{cases} 0\, if\ l\ is\ false : l \rightarrow not\ valid \\ 1\, if\ l\ is\ valid :\ l \rightarrow valid \end{cases}$$

Get all the parameters including the temporary variables and add it into the parameter list.
Let T be the temporary variable.
T is valid only if f(x),f(y),f(z) is in bound
F(t)={ f(x),f(y),f(z) } $\in P$
F(t)=$\int_0^n [\,f(x), f(y)] \in t \rightarrow P(z)$
Since f(t)$\in P$
Let (x,y) $\in P\ o\ f$
X$\in P\ o\ f$ <=> $[\,f(x), f(y)] \in P\ o\ f$
[f(x),f(y)) ∧ *[f (z,y)]* $\in P$ *for some y,z;*
f(x) $\in f \wedge$ (f(x),y) $\in P$ *for some y*
x $\in f \wedge$ f(x) $\in P$
x$\in f(x) \in P$
Hence every temporary variable which has parameter is associated with function.

Create FSM for the parameter list by following the sequential relation from the logic source to terminating point.

If the parameter and the function gives the logic at the runtime the iterative and conditional value are get changed.

Let conditional variable be i={i_1,i_2,i_3,......i_n}
Let the iterative variable be j and k ={ j_1,j_2,j_3...j_n k_1,k_2,k_3.....k_n}

$$F(i) = \begin{cases} 0\, if\ i\ is\ out\ of\ bound \\ 1\, if\ i\ value\ is\ in\ bound\ \ \forall \in P, f, l \end{cases}$$

When i value is between 0 to n

Then add => i value to function {f(x),f(y),f(z)}
Modify the Logic and store in FSM
and if it does not tend to give logic make
i→ *{∅}*
else repeat.

Each state of the simulated FSM stores the parameters and its current value
Find all the parameters are
P(x_1), p(x_2)........p_K ← *q_1* ,q_2,....q_n,t t,r_1,r_2,...,r_m← s_1,s_2,s_3...........s_n

P1, p2....pk, r1,r2,...rm ← *q_1,q_2,...q_1, s_1,s_2,s_3,...s_n*
Theorem: Set of rules is contradictory if and only if another rule can be derived from it.
[proves completeness]
Proof :

Consider the conclusion is not empty.
The value of F, q_1,q_2,q_3....q_l, s_1,s_2,s_3...s_n all have value T and p_1,p_2,p_k , q_1,q_2,......q_l} r_1,r_2,...,r_m all have value F. If t has value f then t, r_1,r_2,...,r_m ← s_1,s_2,s_3...........s_n has value f.

(The value of F, q_1,q_2,q_3....q_l, s_1,s_2,s_3...s_n all have value T)

F(q)={ q_1,q_2,q_3....q_l}, {s_1,s_2,s_3...s_n} $\in T$
(p_1,p_2,p_k , q_1,q_2,......q_l} r_1,r_2,...,r_m all have value F.)
{p_1,p_2,p_k , q_1,q_2,......q_l} } t$\in f$
(If t has value f then t, r_1,r_2,...,r_m ← s_1,s_2,s_3...........s_n has value f.)
If T→ *f(x)*
Then

r_1,r_2,...,r_m← s_1,s_2,s_3...........s_n← f
else

If conclusion is empty then k=l=m=n=0.
Output Parameters <O_1,O_2> are bound with discrete values then return Boolean value (True)

$$B(p) = \begin{cases} 0\, if\ O\ is\ false : l \rightarrow not\ valid \\ 1\, if\ O\ is\ valid\ \ \forall \in x, y, z \end{cases}$$

It proves the Completeness logic.

Algorithm for speed up

```
begin procedure
Define business rule→ Jrule
def rule →string
get the rule
rule← string defining business policy
pass array str[]→ function
input[]→ str [] (rule)
do
//read the rule from rule base
a←input.next()
while(adding→rule)
get the set of conditional rules
        for each rule in the memory()
// locating memory size
        set . memory size→size of memory
//add facts *(rule) to memory
If(old_size=null)
then
        update rule→memory
if(rule.active=true)
search new facts (rules)
        if(conditional rule==positive)
then
        add→new rule
else
no action()
end if
// for deleting rule from memory
        if( size of memory==null)
        set rule active→false
else if
        if (rule active==true)
        validate the rule
end if
end if
end for
end procedure
```

5.PERFORMANCE OF SPEED-UP ALGORITHM

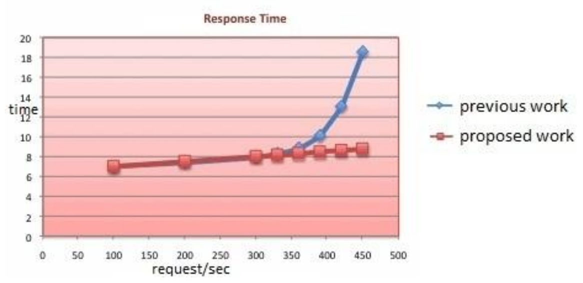

Fig-1. Performance of speed up algorithm

Defines the time taken to response for each rule change request/ second.This graph shows the responsiveness of the rule editor based on the given request. The proposed work responsiveness for the developer's request / sec is less when compared to the previous work.

6. CONCLUSION

This paper provides platform for the business rules to be complete with the help of well defined language. It not only provides completeness for the business rules, but also they could generate definitions of executable rules, processes, and services for the developers. Previous attempts mainly provide rules on the level of concrete syntax without precise language definitions. This system will allow users to model and manage comprehensive rule sets which generate responses in the form of new events. A key focus of this future research work will be the visualization of events which have been processed with sense and respond rules.

7. REFERENCES

[1] Chun Ouyang , "From Business Process Models to Process-Oriented Software Systems"

[2] Michael zur Muehlen, "Business Process and Business Rule Modeling Languages for Compliance Management: A Representational Analysis", Twenty-Sixth International Conference on Conceptual Modeling - ER 2007 - Tutorials, Posters, Panels and Industrial Contributions, Auckland, New Zealand. International Journal on Web Service Computing (IJWSC), Vol.1, No.2, December 2010 .28-31

[3] Josef Schiefer, "Event-Driven Rules for Sensing and Responding to Business Situations".

[4] Claire Costello, " Orchestrating Supply chain Interactions using Emerging Process Description
Languages and Business Rules".

[5] Anca Andreescu, " A General Software Development Process Suitable for Explicit Manipulation of
Business Rules"

[6] Milan Milanovic1, " Modeling Service Orchestrations with a Rule-enhanced Business Process
Language"

[7] Olegas Vasilecas," Ensuring Consistency of Information Systems Rules Models".

[8] Anis Charfi, " Hybrid Web Service Composition: Business Processes Meet Business Rules".

[9] Bruno de Moura Araujo, " A method for Validating the Compliance of Business Processes to
Business Rules".

[10] Nicholas Zsifkov, " Business Rules Domains and Business Rules Modeling".

[11] Antonio Oliveira, " Filho Change Impact Analysis from Business Rules".

[12] Jose F. Mejia Bernal, " Dynamic Context-Aware Business Process: A Rule-Based Approach
Supported by Pattern Identification".

[13] Timon C. Du and Hsing-Ling Chen ,"Building a Multiple-Criteria Negotiation Support System", IEEE transactions on knowledge and data engineering, vol. 19, no. 6, June 2007.

[14] Sam Weber, Hoi Chan, Lou Degenaro, Judah Diament, Achille Fokoue-Nkoutche, and Isabelle
Rouvellou, "Fusion: A System For Business Users To Manage Program Variability", IEEE Transaction
on software engineering , Nov 2008.

[15] H. M. Sneed, "Extracting Business Logic from Existing COBOL Programs as a Basis for
Redevelopment", 9th International Workshop on Program Comprehension, Toronto, Canada, 2001, pp.
167-175.

[16] Sangseung Kang, "Design of Rule Object Model for Business Rule Systems", 1996,pp. 818-822.

[17] Olga Levina, "Extracting Business Logic from Business Process Models", 2010 IEEE.

[18] Mohammed Alawairdhi, "A Business-Logic Based Framework for Evolving Software Systems",
2009 33rd Annual IEEE International Computer Software and Applications Conference.

[19] Olegas Vasilecas, "Ensuring Consistency of Information Systems Rules Models", the International
Multiconference on Computer Science and Information Technology, 2008.

[20] Anis Charfi, " Hybrid Web Service Composition: Business Processes Meet Business Rules",
ICSOC'04, November 15–19, 2004, New York, New York, USA.

[21] Jose F. Mejia Bernal, " Dynamic Context-Aware Business Process: A Rule-Based Approach
Supported by Pattern Identification", SAC'10, March 22-26, 2010, Sierre, Switzerland.

[22] Gulnoza Ziyaeva, Eunmi Choi, and Dugki Min, "Content-Based Intelligent Routing and Message
Processing in Enterprise Service Bus", International Conference on Convergence and Hybrid Information
Technology, 2008

[23] *en*.wikipedia.org/wiki/Business_rule

Authors

Thirumaran. M, working as Asst.Professor in Pondicherry Engineering College, Pondicherry, India, one of India's premier institutions providing high quality education and a great platform for research. He pursued his B.Tech and M.Tech in Computer Science and Engineering from the Pondicherry University. The author is specialized in Web Services and Business Object Model and possesses a very profound knowledge in the same. He has worked on establishing the Business Object Model and Business Logic System with respect to Web Service Computation, Web Service Composition and Web Service Customization. His flair for research has made him explore deep in this domain and he has published more than 20 papers in various International Conferences, Journals and Magazines. Currently he is working on developing a model for Business Logic Systems for various E-Commerce systems.

Ilavarasan. E, working as Associate Professor in Pondicherry Engineering College, Pondicherry, India. He received his Bachelor's degree in Mathematics from the University of Madras in the year 1987 and Master's degree in Computer Applications in the year 1990 from Pondicherry University. Later he completed his M.Tech., degree in Computer Science and Engineeering at Pondicherry University in the year 1997. He has published more than Twenty five research papers in the International Journals and Conferences. His area of specialization includes Parallel and Distributed Systems, Design of Operating System and Web Technology.

Thanigaivel. K, studying in Pondicherry Engineering College, Pondicherry, India. He received his B.Tech degree in Computer Science and Engineering from the Pondicherry University in the year 2009. He currently pursuing M.Tech., degree in Computer Science and Engineering at Pondicherry University and he is currently working in the area of Webservices.