Business Process Requirement Engineering

Atsa Etoundi Roger
Department of Computer Sciences,
University of Yaoundé I,
Yaoundé, Cameroon
roger.atsa@uy1.uninet.cm

Fouda Ndjodo Marcel
Department of Computer Sciences,
University of Yaoundé I,
Yaoundé, Cameroon
marcel.fouda@uy1.uninet.cm

Atouba Christian Lopez
Department of Computer Sciences,
University of Yaoundé I,
Yaoundé, Cameroon
achristianlopez@yahoo.fr

Abstract—**Requirement engineering is as an increasingly important discipline for supporting business process and workflow modeling, as these are designed to satisfy diverse customer needs, and increase the productivity of enterprise. Moreover, most customers hesitate to adopt a given product or service if the added value is not conformed to their desires. Dealing with customers, with a wide range of perspective, within an enterprise, is very complex. These perspectives are grounded in differences in skills, responsibility, knowledge and expertise of stakeholders. This holds more in the domain of business processes and workflows where the satisfaction of the customers is the must if these enterprises wish to deal with the pressure of the network economy. Based on the requirement engineering, we present in this paper an integration of RE approach in the modeling of business process and workflows.** *(Abstract)*

Keywords-component; ***Business Process Modeling, Requirement Engineering, Workflow Management, Business Rule Specification,Business equirement Engineering*** (key words).

I. INTRODUCTION

Business rules are specific policies, standards, practices, regulations, and guidelines that define how employees or enterprise managers carry out business and are therefore considered user-perspective requirements. In the actual functioning of most organizations, based on business requirements, the user requirements are defined. When business rules and user (employees) requirements have been defined, the functional needs are formulated in order to identify the requirements of the expected product. In general, the specification of the requirements and their constraints for the final product is given, based on the functional requirements, as said previously, but also on the data requirements. The data requirements define the specific items that must be included as part of the product. In this requirement process, three levels of requirement are considered [19]: (i) the business level, (ii) the user level, and (iii) the product level.

These three levels of requirement do not take into consideration the customer perspectives which are essential for the production of quality products. However, it is not reasonable to expect sound business or technical decisions during the workflow or business process design on behalf of customers, or to resolve conflicting requirements supplied by different customers, or to set priorities for many requirements that might be collected [21]. For this end, the involvement of the customer is the most critical factor in the production of quality products. The most difficult problem here is sharing the vision of the final product with the customer. We argue that, this problem can be solved if the customers are associated in the production process. This can be done if the business process requirements are driven by the customer requirements that should be taken into consideration for the production of goods or services. These requirements are used in the evaluation of the final product by its beneficiary's i.e customers. Moreover, focusing on the satisfaction of customers and based on the fact that there is a great variety of knowledge within an enterprise that is maintained by various employees, this knowledge should be included in the requirement engineering process from which the business process and workflows are defined. In this work, we give a formal definition business rules and consider that a business rule is a knowledge that help carrying out business. The requirement is considered to be a knowledge that is disclosed in order to meet a certain business goal. This work is an extension of [22] which gives the different levels and types of requirement for the modeling business rule and workflows.

The rest of the paper is organized in five (5) sections. Section two (2) gives the basic concepts that are suitable to model a business rule. In section three (3), we present a formal model of a business rule. In section four (4), relationships between business rules are presented. In section five (5) we shall define the different levels and types of requirements that are suitable in the modeling of a business process in order to deal with the satisfaction of customers and to preserve the enterprise in the global network economy pressure. Section six (6) concludes this paper, and highlights some upcoming works.

II. "BUSINESS RULE" CONCEPT

The "business rule" concept has widely been developed in the context of expert systems. Known by the acronym BRMS ("Business Rules Management Systems"), tools for business rules management are derived from expert systems, which had their heyday in the 1980s [11]. They use similar wordings and all have at the center, an inference engine. Under the business rules-oriented approaches (BROA), this inference engine, is called business rules manager [8]. It aims at modeling human reasoning, behavior and cognitive mechanisms of a human expert in a particular domain. For this, it relies on

a fact base (working memory), an inference engine (rules engine) and a base of rules (production memory) [8]. Facts concepts, inference engine will not be developed as part of this work.

A. Business Rule

In literature, the concept of "business rule" was widely used, each author with his own vision. Thus,

- [15, 16] supports that a business rule is a formulation that defines or constrains some aspect of a business activity. Its purpose is to structure a business activity (policy, know-how), to control or influence the conduct of a business activity;
- meanwhile, [17] says that a "business rule" is a directive, which is meant to influence or guide the conduct of a business activity, in the aim of implementing a business policy that is formulated in response to an opportunity or a risk;
- [4] also thinks that a business rule defines a law of the domain to which the goal must conform to. It helps to organize a management process to achieve the goal. The goal, in turn, defines a potential expectation that the system can satisfy, it expresses what the user of the system wishes to do.

We believe that to better understand the semantics of a business rule, that it is necessary to situate it in a context or specific domain.

Definition (1):
A "business rule" is a directive of a domain which controls the conduct of a business activity of that domain. Its goal is to structure a business activity (policy, know-how) to control or influence the conduct of a business activity of the domain in question, in view of achieving an expected result.

[8] tells us that business rules are divided into two broad categories:

- Structural rules (indicating a necessity): they are rules that addresses how a business activity is organized or structured, the elements comprising it. Structural rules are definition complements;
- Operative rules (indicating an obligation): these are rules that control the manner in which the business activity is carried out. Unlike structural rules, operating rules are those that can be directly violated by business stakeholders.

In the following, we focus exclusively on business rules because they control the business activity.

B. Structure of Business Rules

[8] tells us that, depending on the type of business rule (declarative or procedural), the structure and semantics of that business rule vary. In this paper we focus on production rules. These rules are of the form *"if. . . then. . . [else. . .]"* and consist of the following elements:

- Declarations of variables: used to define the application context of the business rule;
- Salience (Expression) or "level of importance" is an expression whose evaluation returns an integer corresponding to the importance of the rule. It permits us to define an order in the execution of rules. This value can also be called business priority. In [1], we associated the level of importance to knowledge bits. It would be inappropriate to associate it explicitly to the business rule given that this association is implicit in [1]. Thus we have removed this information in the specification that we shall propose in the next section;
- "If" or condition, or "left-hand part", or predicate, or premise (expression): It constitutes the condition under which the rule can be applied; It consists of an expression that is evaluated as true or false. It is also called "body";
- "Then" (side effect expression) or Conclusion or "right-hand side": it is the body of the rule or "head". It represents the action to carry out if the condition part is evaluated as true. Under declarative rules, this part must be an expression without side effects.
- "Else" (optional and side effect Expression). It is only possible for procedural rules if they contain an "If" part. It represents the action to carry out if the condition part is evaluated as false.

Expressions are classified into two groups according to whether they influence the systems' state or not. There are side effect expressions and expressions without side effects. An expression is without side effect if and only if it does not change the state of the environment. Some expressions, on the contrary, during a business activity, do change the state of the environment. They are said to have side effects.

Although in [4, 8, 9, 10, 11, 13, 15, 16], we have a clear vision of what a business rule is, the fact remains that until now, researchers seem not interested in the fact that we can carry out a certain number of operations on these rules. The interest of defining a set of operations on business rules is to be able to:

- reuse business rules in the reengineering of a business process or components-based development.
- compose business rules within the framework of model driven engineering;
- decide on the quality of the formulation of a rule after its specification by experts (business executive of the organization and software experts);
- to open a research branch on the description of a business rule oriented business process;
- evaluate the quality of model composition.

[8] summarizes the formalized specification of a business rule *R* as follows:

$$R = (RuleLabel\ (VariableDeclaration)^*\ (conditions, actions))$$

where:

$$RuleLabel = (how?, Priority?, Visibility?, Refraction?, CreatedTime?, lastModificationDateTime?)$$

$$Conditions = (Expression \mid AndExpression \mid OrExpression \mid NegationExpression)^+$$

*Action = ((Expression) * | NegationExpression^{+})*

In the following, we shall formalize the specification of a business rule as proposed in our approach. We continue subsequently by operations performed on business rules. The term putting objects to the stage, refers to the use of the properties of each object in an expression.

III. PROPOSAL OF A BUSINESS RULES SPECIFICATION

In this section we will introduce the concepts necessary to describe a business rule according to our approach. These concepts are similar to those found in the OCL language [18] on the one hand and on the other hand to those used in BRMS [11]. Generally we maintain the structure of a business rule as defined in [8]. It should be noted in this section that we are not defining a programming language, but a manner of doing, for the specification of business rules, as such this allows us to perform a certain number of operations on these criteria.

A. Predefined Types

1) Simple Types

A simple type ST is given by *(Dt,Nt,Rt,Stg)* where:

- Dt denotes the date type which is used to specify attributes of objects referring to time;
- Nt denotes the number type referring integer values, decimal or real;
- Rt denotes the rule type which is used for the declaration of a business rule;
- Stg denotes the "String" type: referring to a given character or character strings.

A constant a of type Date, Number or String is denoted by "a". Consider a and b two elements of type String, we define the function $\mathrm{SubStr} : \mathrm{String}^2 \rightarrow \mathrm{Bool}$ such that the following properties hold:

$$SubStr(a,b) = \begin{cases} True, & if\ a\ is\ o\ part\ of\ b \\ False, & if\ not \end{cases}$$

In the following, we will denote by $SimpleType$ the set of simple types mentioned above. We also denote by $\prod SimpleType$, the arbitrary choice of a simple type in $SimpleType$.

2) Compound Types

To these four, we add two suffixes: "aggregate", "views", and a complex type. $Something$

The $Something$ type, denoted by *STg*, refers to a physical object. Example: a career management tool, a stamped application, a registration certificate, a national identity card, purchase order, an invoice, etc.. , Whereas:

- the suffix "aggregate" preceded by the keyword "NameSpace", refers to a set of objects in the scope of the rule;
- the suffix "views" before the name of an object in the domain circumscribed by the suffix "aggregate" refers to the set of information (properties or attributes) of this object which shall be used by the business rule.

Objects listed using the suffix "aggregate" can be accompanied by the following entries:

- The citation "control" indicates that the object is obligatory, but no data of this object intervenes in the "description" part of the business rule. If such an element is to be absent, the data is declared invalid. For example in the case of advancement in incremental position, the effective presence is necessary, but it does not affect the actual processing of the said file;
- The mention "reference": it pertains to a value used to uniquely identify any object. When specifying the business rule, this value is not known. A predicate making use of "reference" has the following structure:

$$make\ reference\ to\ \ldots.[in\ \ldots]$$

The mention "artifact" indicates that the object in question is produced by the organization.

B. Declaration of Business Rules

The declaration of a business rule is done in two phases: the definition of its context, and the specification of the domain directives which controls the behavior of the business activity in question. The first part of the declaration of a business rule permits us to specify the usage context of each object of the organization involved in this business activity and the domains in which this business rule can still be used. These domains are listed immediately after the "keywords". Meanwhile the second part is reserved to describing the action that must be done to realize a business. This part is called description.

1) Definition of Context

The specification of any business rule begins by defining its usage context. The usage context of a business rule defines on the one hand, the semantic field of an object *ob* used in the description; and on the other hand, domains in which this business rule may be used (exploitation field of the rule). Formally, we shall define the context of a business rule as follows:

Context:

Keywords =
$Domain_1, Domain_2, \ldots, Domain_n$;
Data =
$NameSpace.aggregate\ \{ob\ \%\ STg : CA$
$[, ob\ \%\ STg : CA]\ *\}$;
$object.views\ \{fd\ \%\ \prod ST\ [, fd\ \%\ \prod ST]\ *\}$

Where:-fd is either a reference to another object which may not be in NameSpace; either a simple name of an attribute or property of that object. fd is also called arguments of the suffix "views" or referenced attribute in the suffix "views".

-$\prod$ ST represents the type of field

- CA = control | artifact

The suffix "aggregate" defines the set of accessible objects when carrying out a business activity for which the rule is being defined. The suffix "views" allows for objects which have no mention "control" to explicitly define the set of information (properties or attributes) of this object, which must be used in the description part of that business rule. A single aggregate is authorized per business rule. Moreover, it is important to note that when an object has the mention "artifact", all its attributes must be defined with the mention "reference" following the syntax defined below. In the following, we shall call context objects, with no mention "control" in the suffix "aggregate". In the following, the concept set will be considered as a MultiSet.

Consider an object b in this context with no mention "control", we shall denote: $\mathrm{card}(b)$, the number of properties of b listed in the suffix « views » ; $\mathrm{NameSpace_C}$, the set of objects referenced in the suffix "aggregate" of context C ; $\mathrm{card}(\mathrm{NameSpace_C})$ represent the number of objects specified in the suffix " aggregate"; $\mathrm{Sg_C}$ the set of signature of objects of context C ; et $\mathrm{Keywords_C}$, the set of words listed after the key word « keyword » in the context C. we denote by $\mathrm{Sig_C}(b)$ the signature of b where $\mathrm{Sig_c}$is defined by $\mathrm{Sig_c}$: $\mathrm{NameSpace_C} \rightarrow \mathrm{Sg_C}$, and

$$\mathrm{Sig_C}(b) = \left(\prod \mathrm{SimpleType}\ \left[,\prod \mathrm{SimpleType}\right]^{*}\right)$$

The signature of b, $\mathrm{Sig_C}(b)$ shows the list of the attributes type defined by the suffix "views".

Given another object a of the same context, we denote$\mathrm{Tri}(a,b)$, the function which returns the signature of a ordered in the order defined by that of b, where Tri is defined by Tri: $\mathrm{NameSpace_C}^2 \rightarrow \mathrm{Sg_C}$, such that for every a and b :

If $\mathrm{card}(a) > \mathrm{card}(b)$, then

$$\mathrm{Tri}(a,b) = \left(\mathrm{Sig_C}(a) \cap \mathrm{Sig_C}(b)\right) \cup \left(\mathrm{Sig_C}(a) - \mathrm{Sig_C}(b)\right)$$

Else,

$$\mathrm{Tri}(a,b) = \left(\mathrm{Sig_C}(a) \cap \mathrm{Sig_C}(b)\right)$$

Axiom 1: Equivalence of *Signature*
(1) We shall say that a and b have equal signatures if and only if:

$$\mathrm{Tri}(a,b) = \mathrm{Sig_C}(b).$$

(2) a and b are said to be equivalent in their context.

Axiom 2: Refinement or Extension of Signature
(3) We shall say that the signature of a is a refinement of b if and only if:

$$\mathrm{Sig_C}(a) \subset \mathrm{Sig_C}(b)$$

(4) We shall also say that b is an extension of a.

Axiom 3: difference of signature

We say that the signatures of objects a and b are different if and only if they satisfy neither axiom 1, nor axiom 2.

Axiom 4: $NameSpace$ and $Contexts$:
Consider two contexts C and B,

(5) $NameSpace_C$ and $NameSpace_B$ are said to be equivalent if and only if:

1. $\mathrm{card}(\mathrm{NameSpace_C}) = \mathrm{card}(\mathrm{NameSpace_B})$ and,
2. $\forall\, a \in \mathrm{NameSpace_B}\ \exists b \in \mathrm{NameSpDce_C}, \mathrm{Sig_B}(a) = \mathrm{Sig_b}(b)$.

(6) We also say that $\mathrm{NameSpace_B}$ refines $\mathrm{NameSpace_C}$ if and only if:

1. $\mathrm{card}(\mathrm{NameSpace_C}) > \mathrm{card}(\mathrm{NameSpace_B})$ and,
2. $\forall\, a \in \mathrm{NameSpace_B}\ \exists b \in \mathrm{NameSpace_C}, \mathrm{Sig_B}(a) = \mathrm{Sig_b}(b)$.

(7) C and B are said to be equivalent if and only if:

1. $\mathrm{NameSpace_C}$ and $\mathrm{NameSpace_B}$ are equivalent ; and,
2. $\mathrm{Keywords_B} \cap \mathrm{Keywords_C} \neq \emptyset$.

(8) We say that the context C refines context B if and only if:

1. $\mathrm{NameSpace_C}$ refine $\mathrm{NameSpace_B}$; and,
2. $\mathrm{Keywords_B} \cap \mathrm{Keywords_C} \neq \emptyset$.

2) Definition of the Description

The "description" part of any business rule is reserved for the specification guidelines that must be carried out to achieve the expected results. Formally, we define the description part of a business rule by the triplet:

$$\mathrm{Description} : (\mathrm{Guard},\ \mathrm{Sequence},\ \mathrm{Results})$$

Where:

- Guard: is the condition to be satisfied in order for actions to be executed;
- Sequence: is a sequence of actions separated by commas;
- Results: is a specification of the expected result.

a. Guard

The guard is a condition defined in [8] which must be satisfied in order for the description part of the business rule to be activated. It is specified by a literal expression of human language featuring objects and attributes listed in the context. Concretely, it is materialized in the form "if condition then ... else ..." to "condition". A guard shall be said to be simple if and only if it is the condition in the formulation "if condition then ...." It is said to be complex if and only if it is the condition in the formulation "if condition then ... else ...".

A condition is an expression of natural language that expresses a reality that can be either true or false. To express expressions, we defined a number of predicates,

of: comparison, coordination, and negation. The following table summarizes the predicates raised.

***Table 1:** list of comparison operators*

Types of predicates	predicates	Notation
comparison predicates	…. Is less than….	(lt, ...,...)
	…. Is less than or equal to….	(le,...,...)
	…. Is greater than….	(ut,...,...)
	…. Is greater than or equal to…..	(ue,...,...)
	…. Is equl to…..	($\stackrel{def}{=}$,... ,...)
Conjunction predicates	…. and ….	($\wedge$,...,...)
	…. or ….	($\vee$,...,...)
Negation predicates	Opposite of….	(¬,...)

Conjunction predicates apply only to conditions. This is a conjunction of conditions using the conjunctions ('or' or 'and'). To reduce misunderstandings between the business executives and developers, it was important to bring our specification closer to natural language. It should be noted however that only objects declared in the context and without the mention "control" should be used. The predicates we define raised three types of expressions: CompExpression reserved for literal expressions depicting two objects of the context associated with a comparison predicate; NegExpression, reserved for expressions expressing the opposite of the reality expressed by a condition. NegExpression Applies only to conditions; CrdExpression, reserved for expressions which translate the composition of conditions. The formal representation of each of the preceding expressions is as follows:

$$\text{CompExpression} = \langle \text{Object}_1.\text{attr}, \text{Object}_2.\text{attr}, \triangle \rangle$$

$$\text{NegExpression} = \langle \text{Cx}_1, \neg \rangle$$

$$\text{CrdExpression} =< \text{Cx}_1, \text{Cx}_2 \mid \text{Cr} \mid \text{NegExpression}, \Diamond >$$

Where:

-$\text{Object}_x.\text{attr}$, represents an attribute of the Object_x, specified in the suffix "views";
- Cx_x is an expression of type *Expression*;
-Cr is an expression of type xpression ;
- $\triangle$ is a comparison predicate;
- $\Diamond$ conjunction predicate;
-'¬' negation predicate.

In general, a condition will be represented formally by:

$$\text{Condition} :< \textit{CompExpression} \mid \text{CrdExpression} \mid \text{NegExpression} >$$

We denote by Conditions, the set of conditions and we define the function $\text{TriCond}: \text{Conditions}^2 \rightarrow \text{Conditions}$ such that for every pair (a, b) of Conditions^2, $\text{TriCond}(a, b)$ returns a sorted condition in the same sense as the above defined Tri function, if the terms a and b are bound by the same conjunction. $\text{TriCond}(a, b)$ returns a in other cases. Consider a condition β, we denote Term(β), the set of predicates comprising the condition β. We shall denote by squelette, the function $\text{Squelette}: \text{Expressions} \rightarrow \text{String}$, which for any expression a, Squelette(a) returns a string obtained by replacing the various attributes of the objects composing this expression by their types without affecting the constants.

Axiom 5: Equivalence of Conditions

Consider two conditions a and b, we say that a and b are equivalent if and only if:

(10) $squelette(a) = squelette(b)$;

(11) one of the squelette in the reverse order of the other.

Axiom 6: Refinement or Extension of a Condition

(12) Consider two conditions a and b, we say that a is a refinement of b if and only if
$SubStr(squelette\ (a), squellete(b))$ is true.

(13) if (10) is satisfied, we say that b is an extension a..

In the following, the notion of squellete will be extended to basic predicates included in the sequences.

b. Sequence

A sequence is a series of actions that must be executed to produce the result of the business activity. An action is a predicate, translating an atomic process or operation for achieving a partial or final objective. It modifies the state of the environment it is an expression with side effects. Two types of actions exist: simple actions and complex ones. Simple actions are those consisted of a single base predicate. Complex actions are those that refer to a set of base predicates, or business rules existing via the deterministic choice operator relative to a condition. The base predicates, inspired by [8], taken into account in our specification are:

- addition of... to ...;
- the withdrawal of .... from ....;
- the product .... by ... ..;
- Division ... .. by ... ..;
- editing of .... on ....;
- recording of .... in ....;
- the creation of ... .. in ....;
- the classification of ... .. in ....;
- the next of ... in ....;
- the previous of ... .. in ....;
- the update .... by ....

The set of base predicates constitutes the domain vocabulary. This concept was well presented in [8]. We will, therefore, not return to. However, it should be noted that extensions of the latter can be done based on natural language and the domain, to cover the set of atomic actions of this human language. Nevertheless, we recommend, basing on tests performed on the

specification of business rules in the field of career management of State personnel and payroll, in a sample of twenty-five administrations, in Cameroon, that we maintain these predicates in state. However the modification of the number of arguments is permitted.

In human language, several twists of language can translate these predicates. It's for the software engineer assisted by the business executive to: specify the business rule, identify the comments of the business executive, the operation in question. To facilitate the expression of the business rule, we have, at the risk of repeating ourselves, chosed predicates which to some extent may reflect the same action. These predicates must put to stage some objects declared in the suffix "aggregate". Besides these basic predicates, we defined a predicate of deterministic choice denoted $[\ldots]\{\ldots\}$, as follows: [obs1,…,obsn]{A1,…,Am} where Ai is in the form actioni(val.obsi,1,…,val.obsi,n), and val.obsi,k represents the value of obsk for the action i, and obsi denotes the property of an object declared in the suffix aggregate.

The business rule action_i shall be executed if and only if the values of the parameters $\text{Obs}_1, \text{Obs}_2, \ldots, \text{Obs}_n$ corresponds to those of $\text{val\_Obs}_{i,1}, \ldots, \text{val\_Obs}_{i,n}$. This predicate enables execution under certain conditions, of a single action from among those listed. It also enables us to represent a complex action of multiple-choice. Formally, the "sequence" will be represented by:

$\text{Sequence} : (\text{SimpleAction} \mid \text{ComplexAction})^+$

where :

-SimpleAction : is an action consisted of an unique basic predicate;

- ComplexAction : $\langle \text{SimpleAction} | [\ldots]\{\ldots\} | [, \text{ComplexAction}] \rangle$.

We denote by Sequences, the set of business process sequences.

Axiom 7: Equivalence of actions

Two actions a and b are said to be equivalent if and only if :

$$\text{squelette}(a) = \text{squelette}(b).$$

We define the function processStr as follows:

$\text{processStr} : \text{Sequences} \times \text{NameSpace}_C \rightarrow \text{SimpleType}$ such that $\forall (s, o) \in \text{Sequences} \times \text{NameSpace}_C C$, $\text{processStr}(s, o)$ is the set of processes which object o is subjected to in the sequence s. $\text{processStr}(s, o)$ is also called the modifications string of the object o. $\text{processStr}(s, o)$ materializes the effect of processes on the object o.

Axiom 8: Equivalence between Sequences

Consider two sequences s_1 and s_2, we say that s_1 and s_2, are equivalent if and only if : $\forall a \in \text{NameSpace}_C$, $\text{processStr}(s_1, a) = \text{processStr}(s_2, a)$

Axiom 9: Extension, Refinement of Sequence

Consider two sequences s_1 and s_2, we say that s_1 is a refinement of s_2 (or s_2 is an extension of s_1) if and only if the s_2 contains the predicate of deterministic choice and there exist a sequence s in this predicate, such that, if a is an element of NameSpace_C, then

$$\text{processStr}(s_1, a) = \text{processStr}(s, a)$$

c. Results

The result is what we observe at the end of the business activity. Results indicators are defined in the same way as for attributes of object that are used in the description part of a business rule. In a formal way, let *Ob* be an object, *F* a given field of *Ob* of a given simple type *ST*, a result *Rt* is defined as follows:

$$Rt = (Ob.views\{\ F\ \%ST[, F\%ST]^*\})^+$$

Meanwhile, in the result specification of results, certain attributes of objects may not contain the mention "reference". We denote by $ResultObjets$, the set of results objects of a business rule and $card(ResultObject)$ the number of objects in this set. Elements of $ResultObjets$ implicitly have the mention "artifacts." The results can also be seen as the goal to attain.

Axiom 10: Equivalence Results

Consider two results a and b, we denote ResultObjets_a (ResultObjets_b respectively), the set of result objects of a (of b respectively) we shall say that a and b are equivalent if and only if:

1- $card(ResultObjets_a) = card(ResultObjets_b)$ and,

2- $\forall o_1 \in ResultObjets_a$,
$\exists o_2 \in ResultObjets_b, Sig_a(o_1) = Sig_b(o_2)$.

Axiom 11: Extension, Results Refinement

Consider two results a and b, we denote ResultObjets_a (respectively ResultObjets_b) the set of result objects of a (of b respectively), we say that a is a refinement of b, if and only if:

1- $\text{card}(\text{ResultObjets}_a) > \text{card}(\text{ResultObjets}_b)$ and,

2- $\forall o_1 \in \text{ResultObjets}_a$,
$\exists o_2 \in \text{ResultObjets}_b, \text{Sig}_a(o_1) = \text{Sig}_b(o_2)$.

After presenting the different parts of a business rule, we formally define the business rule as follows:

$\text{rule\_name} = (\text{Contexte}_{\text{rule}}, \text{description}_{\text{rule}})$

where: - $\text{Contexte}_{\text{rule}}$: represents the context part of the business rule

- $\text{description}_{\text{rule}}$: represents the description part of the business rule

The formal representation of the context and description parts of a business rule were given in the previous sections. We also defined a number of relationships such as equivalence, refinement between concepts developed in the sections above. As we proceed, we shall use these relations to develop relationships between the business rules.

IV. RELATIONSHIP BETWEEN BUSINESS RULES

In the previous section, we have defined a set of concepts and relationships between these different concepts. All these works were intended to clearly present our vision of the business rule and prepare the ground for defining relationships between different business rules. The lines that follow shall be devoted to these relationships. Furthermore, we denote equivalence between two concepts by: ≡; refinement by: ≅, and $views(b, \alpha)$, all attributes of the object b in the business rule α.

A. Axioms

Consider two business rules α and β, we denote by Contexte_α, Contexte_β the respective contexts of the business rules α and β and $\text{Description}_\alpha = (\text{Garde}_\alpha, \text{Sequence}_\alpha, \text{Result}_\alpha)$, $\text{Description}_\beta = (\text{Garde}_\beta, \text{Sequence}_\beta, \text{Result}_\beta)$ the descriptions part of the business rule α and β respectively.

Axiom 12: Equivalence of Business Rules

We say that α and β are equivalent if and only if:

$$\text{Contexte}_\alpha \equiv \text{Contexte}_\beta \text{ et } \begin{cases} \text{Sequence}_\alpha \equiv \text{Sequence}_\beta \\ \text{Garde}_\alpha \equiv \text{Garde}_\beta \\ \text{Result}_\alpha \equiv \text{Result}_\beta \end{cases}$$

Axiom 13: Extension, Refinement of Business Rules

We say that α is a refinement β or that β is an extension of α if and only if:

$$\text{Contexte}_\alpha \cong \text{Contexte}_\beta \text{ and } \begin{cases} \text{Sequence}_\alpha \cong \text{Sequence}_\beta \\ \text{Garde}_\alpha \cong \text{Garde}_\beta \\ \text{Result}_\alpha \cong \text{Result}_\beta \end{cases}$$

Axiom 14: Inconsistent Business Rules

We say that a business rule is inconsistent, if and only if:

1- there exists at least one object in the context of this business rule that is not used in the description of that business rule;
2- there exists at least one attribute or property of an object from its context which is not used in the description part of that business rule;

Axiom 15: Incomplete Business Rules

A business rule is said incomplete if and only if it is not inconsistent and there are properties that have the mention "reference" which does not appear in the result objects of the same rule.

Axiom 16: Merging Business Rules

We say that two business rules α and β can merge if and only if:

1- α and β are neither inconsistent nor incomplete;
2- $\text{Keywords}_\alpha \cap \text{Keywords}_\beta \neq \emptyset$ and $\text{NameSpace}_\alpha \cap \text{NameSpace}_\beta \neq \emptyset$
3- $\exists\, b \in \text{NameSpace}_\alpha \cap \text{NameSpace}_\beta$ and $\text{views}(b, \alpha) \cap \text{views}(b, \beta) \neq \emptyset$;

B. Impact of these Relations on the Business Process Requirements Model

Definition (2)**:** Sequencing Rule

A business rule α is a sequencing rule if Sequence_α contains the predicate of deterministic choice.

In [1], we presented a goal oriented approach- for the definition of a business process requirement model, taking into account their level of importance and constraints inherent to these requirements. The level of importance of a goal is the credit which the user associates to this goal. Constraints are non-functional requirements related to what this goal must satisfy. The approach that was proposed in [1], revolves around four main activities: requirement elicitation, selection of different goals, transformation of requirements into knowledge bits and finally the development of the requirement model. We have shown formally that this approach will exhaustively describe a business process. To do this, we have given formalism to model the requirements of a business executive and deduced from the work of [4], a formal representation of what we call knowledge bit or expressed requirement. An expressed requirement or knowledge bit was defined as follows:

$$\partial = (\psi, \omega, \lambda, \delta, \nu)$$

where :
∂ name of knowledge bit
ψ is the context in which the goal is defined
ω is the goal
λ is the business rule
δ represents contraints
ν is the level of importance of the goal
∂ is the name of a domain concept ψ.

Let's consider $a = (\psi, \omega, \lambda, \delta, \nu)$ and $b = (\psi', \omega', \lambda', \delta', \nu')$ two expressed requirements S^a is the set of objects of the organizations' information system, for which the expectation $a.\omega$ is satisfied under the rule $a.\lambda$ and the constraint $a.\delta$ [1]. It is the same for S^b

The concepts of requirements identity, sub-requirements, and sub division of requirements were clearly defined in [1]. This definition was exclusively focused on usage intension. We shall not return to this. We shall use these characteristics to show the impact of a business rule on the organizations' business processes requirement model.

Impact 1: $S^a = NameSpace_{a.\lambda}$ by definition.
Impact 2*:* $a.\lambda \equiv b.\lambda'$ therefore a and b are identical

Proof: We assume $a.\lambda \equiv b.\lambda'$ and shall show that $\psi = \psi', \omega \approx \omega' \; et \; S^a = S^b$

Consider two requirements a and b, in the conditions of the paragraph above, we assume that $a.\lambda \equiv b.\lambda'$, by definition of $a.\lambda \equiv b.\lambda$ we have:

a) $Contexte_{a.\lambda} \equiv Contexte_{b.\lambda'}$, that is $NameSpace_{a.\lambda} \equiv NameSpace_{b.\lambda'}$ *and* $Keywords_{a.\lambda} \cap Keywords_{b.\lambda'} \neq \emptyset$, *hence,* $S^a = S^b$;

b) $Results_{a.\lambda} \equiv Results_{b.\lambda'}$, by definition, $\omega \approx \omega'$.

c) From a) and b) we deduce that $\psi = \psi'$.

Impact 3 : $a.\lambda \cong b.\lambda'$, a is a sub-requirement of b.

Proof: Suppose that $a.\lambda \cong b.\lambda'$. 'and show that $S^a \subset S^b$ and $b = \rho^a$. Consider two requirements a and ; by $\lambda \cong \lambda'$ in the conditions in the above paragraph, we assume that $a.\lambda \cong b.\lambda'$; by definition of $a.\lambda \cong b.\lambda'$ we have: $\lambda \cong \lambda'$ definition of a:

a) $Contexte_{a.\lambda} \cong Contexte_{b.\lambda'}$, that is to say $NameSpace_{a.\lambda} \cong NameSpace_{b.\lambda'}$ and $Keywords_{a.\lambda} \cap Keywords_{b.\lambda'} \neq \emptyset$. $NameSpace_{a.\lambda} \cong NameSpace_{b.\lambda'}$ we deduce by definition that $S^a \subset S^b$;

b) $Séquence_{a.\lambda} \cong Séquence_{b.\lambda'}$ by definition the $a.\lambda$ is referenced in $b.\lambda'$ therefore $b = \rho^a$.

Impact 4: (definition of divisible requirement):

We say that a is divisible if and only if $\text{Sequence}_{a.\lambda}$ contains the predicate of deterministic choice. The number of business rules referenced in the predicate of deterministic choice constitute the number of parts of requirement a.

Impact 5 (definition of ambiguous requirement):

We say that an expressed requirement a is ambiguous if and only if $a.\psi$ is not in the list of domains listed in the domain of operations of the business rule.

Impact 6 (definition merging requirements*):*

We shall say that two requirements a and b can merge if and only if $a.\lambda$ is merged to $b.\lambda$.

These impacts allow us to complete the definition of the concepts discussed in [1] which remained superficial.

C. Business Object Model (BOM)

We mean by "Business Objects", an object referenced in a business rule. Business objects are manipulated in the description part of a business rule. This part enables one to describe exhaustively the various business objects of business processes. In this section, we present the methodology of defining business objects.

Consider $R_{BP} = \cup_i^n b.\lambda$ (where b is an expressed requirement, λ the business rule of b), the set business rules of a business process $\text{NameSpace}_{BP} = \cup_{i=1}^n \text{NameSpace}_{r_i}$ (where r_i is the ith business rules; NameSpace_{r_i} is the set of objects referenced in the rule r_i), of objects referenced in all business rules; A_{BP}, all the business objects attributes, a function prop: $\text{NameSpace}_{BP} \times R_{BP} \to A_{BP}$ that for every pair (a, r) of $\text{NameSpace}_{BP} \times R_{BP} \to A_{BP}$, $\text{prop}(a, r)$ returns the set of attributes of the object a referenced in the rule r .We define a function fields : $\text{NameSpace}_{BP} \to A_{BP}$, such that if $b \in \text{NameSpace}_{BP}$, then

$$\text{fields(b)} = \bigcup_{r \in R_{BP}} \text{prop(b, r)}.$$

Property 1: Business Object Attributes

Consider a business object O, of NameSpace_{BP}, fields represents the set of attributes of the object O.

Let fieldref denotes a function *field^{ref}: NameSpaceBP* $\to A_{BP} \times NameSpace_{BP}$, the attributes of an object $b \in$ NameSpaceBP referenced in a business rule is given by :

$$field^{ref}(b) = \bigcup_{r \in R_{BP}} prop^{ref}(b, r).attributs_b$$

Where:

- prop^{ref}: $\text{NameSpace}_{BP} \times R_{BP} \to A_{BP} \times \text{NameSpace}_{BP}$ is a function for every pair (b, r) of $\text{NameSpace}_{BP} \times R_{BP}$, $\text{prop}^{\text{ref}}(b, r)$ returns the set of pairs $(\text{attribut}_b, a)$ such that attribut_b represents the set of attributes of the object b having the mention "reference" in the suffix "views" of the object of a business rule r.

Property 2: *Reference Attributes of a Business Objects*

Consider a business object O of NameSpace_{BP}, $\text{fields}^{\text{ref}}(O)$ represents the set of attributes of the object O with the mention "reference" in the suffix "views" of business objects of NameSpace_{BP}.

Let $fields^{tag}$, be a function $NameSpace_{BP} \to A_{BP} \times NameSpace_{BP}$, the attributes of an object b referenced in the business rule is defined by :

$$fields^{tag}(b) = \bigcup_{r \in R_{BP}} prop^{ref}(b, r).a$$

Where:

- $prop^{ref}$: $NameSpace_{BP} \times R_{BP} \to A_{BP} \times NameSpace_{BP}$ is a function for every pair (b, r) of $NameSpace_{BP} \times R_{BP}$, $prop^{ref}(b, r)$returns the set of pairs $(attribut_b, a)$, such that attribut_b represents the set of attributes of the object b having the mention "reference" in the suffix "views" of the the business rule object r.

Property 3: Links between business Objects

Consider a business object O of NameSpace_{BP}, $fields^{tag}(O)$ represents the set of business objects with references in the attributes of O. These references are called links or connections between O and other business objects.

Constraints*: (uniqueness of the name business object)*

The name of a business object is unique in a business processes. It can be reused as many times as you want in the context part of different rules. It is the same for attributes of these business objects. The latter retains their types and their names, whatever the business rule.

V. LEVELS AND TYPES OF REQUIREMENTS FOR BUSINESS PROCESS AND WORKFLOW MODELING

In the previous sections; we have defined an approach to model business rules and its constraints. The defined model is based on the concept of context, action and conditions which are essential for the definition of knowledge within an organization. We then consider in this section that a model of a business rule defines specific knowledge which is disclosed for the achievement of a business goal.

A business goal is achieved within an organization by considering not only actions to be taken by employees, but also the needs of customers for their satisfaction as all the products, goods or services produced within an enterprise are designed for them. For this end, specific attention should be taken in order to ensure that the output of the process will be accepted and help the associated enterprise dealing with the global network economy pressure [20].

In the daily life of an enterprise, some employees are breaking out of the process while others are entering the process. This mobility of employees has a certain effect in the delivery of qualified services to customers as the new comers do not have, most of the time, the necessary skills or experiences to deal with activities related to their positions. This concern should be addressed when dealing with the design of a workflow within an enterprise as not only machines are participating in the achievement of activities, but some parts of these activities are carried out by humans. Each human has a tacit and public knowledge, and when he breaks out of the process, if nothing is done, he goes with this knowledge. As a consequence, the enterprise has to train new employees in order to maintain the defined quality of service. However, within the capacity building period, the quality of service is not as good as expected and the beneficiaries of these services, most of the time, look for other enterprises for the delivery of qualified services. Therefore, in the process of defining the requirement for the design of a workflow, the management of knowledge is something that should be considered [21].

In our approach in defining requirements for the design of a business process and a workflow, four perspectives are to be considered, (1) the customer perspective which defines their perception of the quality of service and the required quality of service requirement, (2) the business perspective where the business requirements are defined based on requirements defined in the customer level, (3) the employee perspective employee requirements are defined in terms of skills and experiences, but also the business rules needed to perform various activities based on the knowledge gathered during the previous performances of activities, and finally (4) the product perspective where the requirement of the end product are given based on the previous perspectives. These four perspectives are given by the figure 1 below.

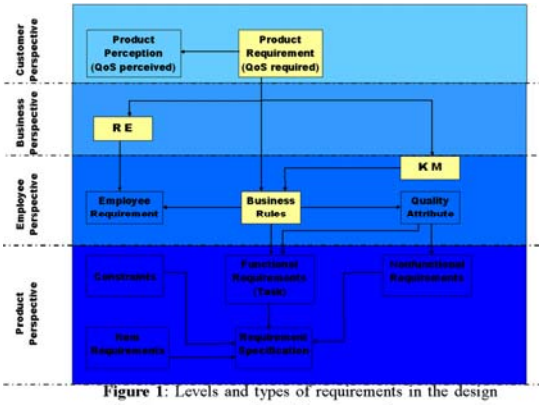

Figure 1: Levels and types of requirements in the design of business process and workflow perspective

VI. CONCLUSION AND FUTURE WORKS

Dealing with the satisfaction of customers is a challenge that enterprises have to tackle in order to resist the pressure of the network economy. Enterprises that fail to take into consideration this challenge will go bankrupt as their products are no more accepted by various consumers. To help enterprises in producing qualified products, several methods have been defined for the formalization of their business processes and workflows. However, despite the popularity of these methods, little has been done in the integration of various requirement types and perspectives in the modeling of business process and workflows.

In this paper, we have defined a way of formalizing the business process rules that we considered to be the key concept in the integration of a variety of knowledge in the model of business process and workflow to increase the productivity within an enterprise and the satisfaction of the end users. After its formalization, a business process rule has been considered to be a set of knowledge that is used to carry out business by both parties' enterprises and customers. Based on this formalization, we have defined four perspective levels for the design of a business process and a workflow within an enterprise. These perspectives include, (i) the customer perspective, where the requirements regarding the quality of service desired are defined, also the knowledge concerning their perception of the service delivered, (ii) the business perspective where the requirements regarding the structure of the business process are defined, (iii) the employee perspective where the requirements concerning the skills and experiences are defined and also the knowledge used in the processing of activities based on the knowledge gathered during the previous processing steps, (iv) the

product perspective where the requirements regarding the specification of the final product are defined.

At this state of our work, we do not deal with the explicit management of knowledge within an enterprise or the selection of knowledge as not all the knowledge is required to be taken into consideration. This work does not also go deeper in the modeling of customers for the formal definition of their perception and their degree of importance given different criteria. These topics are some of future works that can be carried out for the refinement of the proposed approach.

REFERENCES

[1] R. Atsa Etoundi, M. Fouda Ndjodo, Christian Lopez Atouba, "A Goal Oriented Approach for the Definition of a Business Process Requirement Model", IJCA, 2010.

[2] R. Atsa Etoundi, M. Fouda Ndjodo, « Human Resource Constraints driven Virtual Workflow Specification », IEEE SITIS pp 176-182, 2005.

[3] R. Atsa Etoundi, M. Fouda Ndjodo, « Feature-Oriented Business Process and Workflow », IEEE SITIS pp 114-121, 2005.

[**4**] Farida Semmak, Joël Brunet, « Un métamodèle orienté buts pour spécifier les besoins d'un domaine », 23e Congrès INFORSID, pp 115-132, mai 2005.

[**5**] Lubars, M., Potts, C., Richer, C.: A review of the state of the practice in requirements modeling.Proc. IEEE Symp. Requirements Engineering, San Diego 1993.

[**6**] Karen Mc Graw, Karan Harbison, User Centered Requirements, The Scenario-Based Engineering Process. Lawrence Erlbaum Associates Publishers, 1997.

[**7**] The Standish Group, Chaos. Standish Group Internal Report, http://www.standishgroup.com/chaos.html, 1995

[**8**] Mouhamed Diouf, « Spécification Et Mise En Œuvre D'un Formalisme De Règles Métier» , thèse n°3507, Université Bordeaux I, décembre 2007.

[**9**] "Anonyme" Business Semantics of Business Rules, *Business Rules Team Response to RFP: SBVR Submission bei*/2005-08-01 Version 8. http://www.omg.org/docs/bei/05-08-01.pdf last accessed on 2007/05/02

[**10**] "Anonyme" Semantics of Business Vocabulary and Business Rules Specification, Document de specification de l'OMG, 2006. http://www.omg.org/docs/dtc/06-03-02.pdf last accessed on 2007/05/02

[**11**] Vincent Legendre, Gérald Petitjean , Thierry Lepatre, « Gestion des règles métier », Génie Logiciel ♦ n° 92 mars 2010, pp 43-52, 2010

[**12**] Tim van Eijndhoven, Maria-Eugenia Iacob, María Laura Ponisio "Achieving Business Process Flexibility With Business Rules", 12th IEEE EDOCC, pp 95-104,2008.

[**13**] Object Management Group, *Production Rule Representation: Request for Proposal*, br/2003-09-03, Sept. 2003. http://www.omg.org/docs/br/03-09-03.pdf.

[**14**] David Hay and Keri Anderson Healy. De_ning Business rules What Are They Really ? Technical Report 1.3, The Business Rules Group, July 2000

[**15**] Ronald G. Ross. Principles of the Business Rule Approach. Addison-Wesley, Boston, USA, 2003.

[**16**] Business Rule Group Community BRG. http ://www.brcommunity.com. Web page.

[**17**]D. Hay and K. A Healy. "GUIDE Business Rule Project Final Report". Technical report, 1997.

[18] The Object Management Group OMG, UML 2.0 OCL Specification.OMG Specification, October 2000

[19] Karl E. Wiegers, In Search of Excellent Requirements, Journal of the quality Assurance Institute, January 1995.

[20] Schahram Dustdar, "Reconciling knowledge management and workflow management system: the activity-based knowledge management approach"', Journal of Universal Science, vol.11, no. 4, 2005.

[21] Lynette L. Ralph and Timothy J. Ellis "an investigation of a knowledge management for the improvement of reference services", journal of information, information technology, and organizations, volume 2, 2009

[22] Atsa Etoundi Roger, Fouda Ndjodo Marcel and Atouba Christian Lopez. Article: *A Model based Business Process Requirement Rule Specification*. International Journal of Computer Applications 11(9):17–24, 2010.