Towards an Approach for Estimating Impact of Changes on Business Processes

Mohamed Boukhebouze[1], Youssef Amghar[1], Aïcha-Nabila Benharkat[1], and Zakaria Maamar[2]

[1]*Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France*
{mohamed.boukhebouze, youssef.amghar, nabila.benharkat}@insa-lyon.fr
[2]*College of Information Technology, Zayed University, Dubai, UAE*
zakaria.maamar@zu.ac.ae

Abstract

A business process needs to be constantly reviewed to accommodate new business requirements and regulations. To this end, it is important to manage the impact of this review on a process by determining which parts are affected and more critically estimating the overall cost of this review. In this paper we present an approach to manage the changes in a process. We model a process as a set of business rules that are structured using the ECAPE formalism standing for Event, Condition, Action, Post–condition, and post-Event This formalism allows translating a process into a graph of rules that is used for two types of analysis: business processes agility and cost of changes. This cost is based on our Rule Change Cost Model (R2CM).

1. Introduction

In today's business world, organizations have to manage complex processes and quickly react to frequent changes. A business process needs to be constantly updated to support new business requirements and regulations. According to Goedertier et al. the origin of change comes mainly from frequent changes in first, regulations that organizations have to comply with and second, internal policies that organizations themselves develop [1]. These regulations and policies are often expressed in terms of business rules that are sometimes defined using high-level structured statements that constrain, control, and influence business logics [2].

Business rules should be formalized to facilitate their understanding, validation, and then use. Unfortunately, using imperative languages (e.g., BPEL [3]), designers implement business rules using decisions (what process branch should be chosen) that are defined using connectors (e.g., sequence, parallel split, exclusive choice). Designers use the results of the decisions to determine the process behavior rather than to model these decisions. This makes business processes rigid, which leads to a costly change effort. Indeed, to implement a change in some parts of a business process (e.g., adding a task, removing a task, or modifying a constraint), designers should re-examine the entire process model. However, it would be beneficial to modify only the parts that are affected while keeping other parts of the process untouched. Moreover, experiments have shown that organizations express their policies and regulations in business rules using natural language or adding text annotations to their models [4]. However, the formulation of business rules must be rigorous, concise, and accurate to ensure that these rules are unambiguous and coherent.

In our previous work [5], we proposed a rule-based approach to model the logic of a process with a set of business rules that comply with declarative languages' guidelines. This way of doing allows deploying partially-specified process definitions [6]. Indeed, a rule engine determines, at runtime, what to execute by evaluating relevant rules with regard to a certain process event. According to Lu et al., a rule-based approach externalizes the process logic from the execution environment [6]. Consequently, the modifications in a process definition can be done without impacting the executing process instances. In addition, the changes (in process logic, business regulations, or business policies) are implemented by changing a subset of rules (e.g., modify, insert and delete existing rules), which express the changed process logic, the changed business regulations or the changed business policies. As a result, the modification in a rule impacts only a subset of rules that are related to the changed rule, which would lead to a reduction of the efforts to put into this change management.

However, when it comes to complex processes, it is important to manage the impact of a rule change on the rest of the process by determining which rules are impacted by this change and estimating the overall cost of this change. Although, the evaluation of business process changes impact is not trivial and should be carefully examined, this evaluation is beneficial when several change alternatives are offered and planning, organizing, and managing resources to ensure these changes success.

In this paper, we develop an approach that can manage the flexibility of business process rule based modeling by estimating the impact and the cost of business process changes. By flexibility we mean how to implement changes in some parts of a business process without affecting the rest of parts neither the continuity and stability of these parts [7]. The research questions that are raised here concern: what is the rule formalism that would offer better support to change management impact, and how is this change impact estimated? We propose a new rule based model that extends ECA rules and is built upon formal tools. The business logic of a process is summarized with a set of rules that implement an organization's policies. Each business rule is formalized using our ECAPE formalism (Event/Condition/Action/Post-condition/ post-Event). An ECAPE process can be easily translated into a graph of rules. This permits to verify if such a graph guarantees flexibility by studying the relationships between the rules and assigning cost estimation to rule change by using our change cost model called **R**ules **C**hange **C**ost **M**odel (R2CM).

The rest of this paper is organized as follows. We introduce in section 2 our ECAPE rules modeling for business process. In section 3, we detail our approach for process flexibility management and impact change estimate. We wrap up the paper with related work, conclusion, and some directions for future works.

2 Rule based modeling of business processes

2.1. Definitions

The objective of a rule based model is to describe business processes using a set of connected rules. As a result, a sequence of rules define the behavior of a process. According to Giurca et al. in [8], it is advantageous to use reactive rules (ECA formalism) for specifying business processes. Giurca et al. justify that rules give a flexible way to specify a process control flow by using events. In addition, ECA rules are easier to maintain and cover other kinds of business rules (integrity rules, which concerns the assertions that must be satisfied, derivation rules, which explain one or more conditions and one or several conclusions). However, a new type of ECA formalism that helps control the execution of the set of rules is required. To this end, we propose the formalism ECAPE which is defined as:

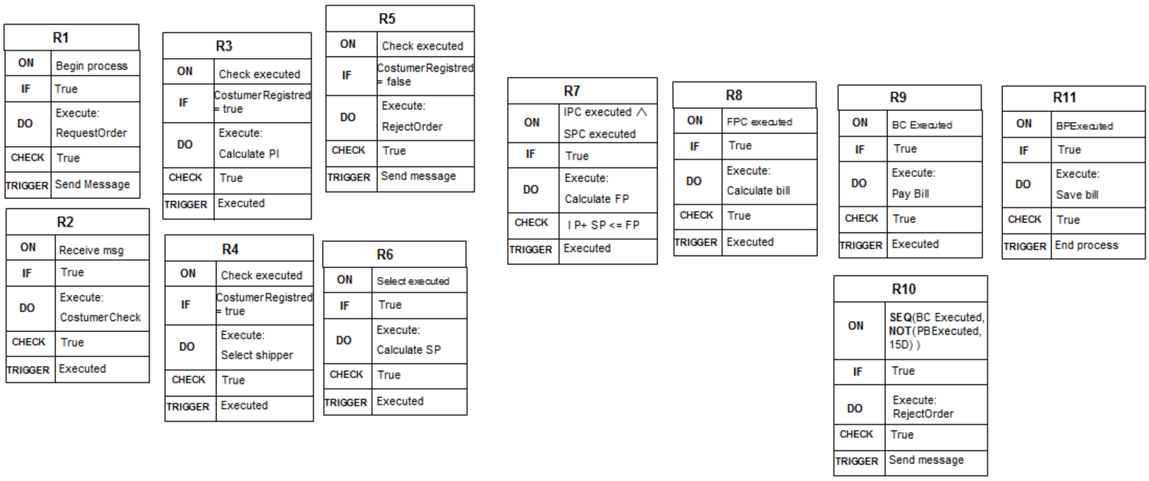
```
ON      <Event>
IF      <Condition>
DO      <Action>
Check   <Post condition>
Raise   <post Event>
```

The semantics attached to an ECAPE rule is: event determines when a rule must be evaluated (or activated); condition is a predicate upon which the execution of action depends (it can be seen as a refinement of the event); action specifies the code to execute if the condition is true; post-condition is a predicate upon which the validation of the rule depends (the rule is validated only if the post condition is true); and event-triggered (post events) identifies the set of events that arise after the execution of the action. Note that, if a post condition does not hold, a compensation mechanism is launched in order to try, if possible, to cancel the executed action's effects. Compensation mechanisms do not fall within the scope of this paper.

A sequence of ECAPE rules defines the behavior of a process. Each rule may activate one or more rules. The originality of this formalism is that the set of events triggered after the execution of a rule's action, is explicitly described. As a result, a sequence of rules can be automatically deducted.

2.2 Illustrative example

In this section we introduce the famous example of purchase order process to illustrate our proposed formalism. Upon receipt of a customer order, the calculation of the initial price of the order and shipper selection are done simultaneously. When both tasks are complete, a purchase order is sent to the costumer. In case of acceptance, a bill is sent to the customer back. Finally, the bill is registered. Two constraints exist in this scenario: customer record must exist in the company database, and bill payment must be done 15 days before delivery date. Figure 1 represents the ECAPE rules set of the purchase order process. In the new business process model, a process is seen as a set of decisions and policies. Both are defined by a set of business rules. For example, rule R1 expresses the policy of requesting an order. This rule is activated by "begin process" event that represents customer order (it may be, for example, click on "Place Order" button). The execution of "RequestOrder" activity triggers "Send message" event. The latter will activate rule R_2 that expresses the policy of receiving an order. Upon "Receive Order" event occurrence, the rule is triggered and the action's <Execute> instruction is performed. This instruction specifies that a given business activity must be performed ("CostumerCheck" in our example). The execution of this instruction triggers "Costumer Check Executed" event. The latter activates three rules R_3 (policy for initial price calculation), R_4 (policy for shipper selection) and R_5 (policy for order rejection when costumer is not registered). The execution of these rules' actions activates other rules.



R1	
ON	Begin process
IF	True
DO	Execute: RequestOrder
CHECK	True
TRIGGER	Send Message

R2	
ON	Receive msg
IF	True
DO	Execute: CostumerCheck
CHECK	True
TRIGGER	Executed

R3	
ON	Check executed
IF	CostumerRegistred = true
DO	Execute: Calculate PI
CHECK	True
TRIGGER	Executed

R4	
ON	Check executed
IF	CostumerRegistred = true
DO	Execute: Select shipper
CHECK	True
TRIGGER	Executed

R5	
ON	Check executed
IF	CostumerRegistred = false
DO	Execute: RejectOrder
CHECK	True
TRIGGER	Send message

R6	
ON	Select executed
IF	True
DO	Execute: Calculate SP
CHECK	True
TRIGGER	Executed

R7	
ON	IPC executed ∧ SPC executed
IF	True
DO	Execute: Calculate FP
CHECK	I P+ SP <= FP
TRIGGER	Executed

R8	
ON	FPC executed
IF	True
DO	Execute: Calculate bill
CHECK	True
TRIGGER	Executed

R9	
ON	BC Executed
IF	True
DO	Execute: Pay Bill
CHECK	True
TRIGGER	Executed

R10	
ON	**SEQ**(BC Executed, **NOT**(PBExecuted, 15D))
IF	True
DO	Execute: RejectOrder
CHECK	True
TRIGGER	Send message

R11	
ON	BPExecuted
IF	True
DO	Execute: Save bill
CHECK	True
TRIGGER	End process

Figure 1: ECAPE rules set of the purchase order

Note that, for the validation of rule R_7, the post condition is satisfied (initial prices + shipping price <= final price is true). Another point is that, the event that activates rule R_{10} is a composite event. For this reason, we should use the constructor event SEQUENCE and NOT to express the fact that the rule is activated when the bill calculation is executed, after that, any occurrence of "pay bill executed" event is happened until 15 days. However, taking into account the dynamic of the various process elements made us think of a better model for business process modeling. For example, if the enterprise decides not to deliver its products, rule R_4 will be deleted from the process model with minimal impact on the rest of rules. Consequently, we deem appropriate to determinate the set of rules that would be impacted by the change of rule R4 in order to keep the coherence of the process and estimate the overall cost of this change.

In the next section we detail our approach to manage the flexibility of rule based business process modeling and estimate the impact and cost of business process changes.

3. Change management

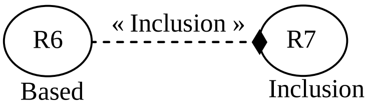
In our work we aim at automating the management of the flexibility of business process rule based modeling by estimating the impact and cost of business process changes. This should help in planning, organizing, and managing the necessary resources that would achieve these changes. This estimation is beneficial when change alternatives are offered. Our approach is as follows.

Firstly, we study the relationship between the rules to determinate the rules that will be affected by a change in a certain rule. Secondly, we formalize the flexibility management of the process modeling by translating the process into a graph of rules. Thirdly, we determine the impact of a rule change by using an associated algorithm. Finally, we estimate the overall change cost by using our ***R**ules **C**hange **C**ost **M**odel* *(R2CM)* that takes into account two parameters: the nature of the relationships between rules and the rule distance in the graph of rules.

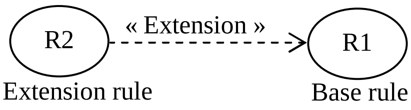
3.1 Relationships between business rules

A change in a process element may require changing other elements that are related to this element for the sake of process consistency. Therefore, we need to study the relationships between the business rules in order to automate the flexibility modeling management. We identify three relationships between rules:

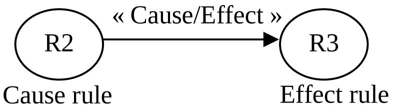
1. Inclusion relationship: shows the case of a rule (base rule) that includes the functionality of another rule (inclusion rule). Two rules have an included relationship between them if the completion of the base rule's action requires the completion of the inclusion rule's action. In the previous example, to calculate the final price, the shipping price must be calculated before.



2. Extension relationship: shows the case of a rule (extension rule) that extends the functionality of another rule (base rule). Two rules have an extension relationship between them if the completion of the extend rule's action achieves the completion of the base rule's action. In the previous example, if we suppose that a loyal customer receives a discount and a new discount rule R_{12} is added. As a result, there is an extension relationship between R_2 (rule to identify a costumer) and R_{12} (rule to calculate discount) because the functioning of R_2's action will complete the functioning of R_{12}'s action.



3. Cause/Effect relationship: shows the case of a rule (cause rule) that activates another rule (effect rule). Two rules have a cause and effect relationship between them if the execution of a rule will activate the effect rule. As a result, the execution of a cause rule's action triggers a post event, which necessarily activates the effect rule. Thanks to this relationship, the order of process activities can be defined by describing the post events based on ECAPE. In our previous example, the performance of R_2's action (verify costumer) will trigger end-customer–verification post-event. This latter is the event activator of rule R_3. There is a cause and effect relationship between R_2 and R_3.



Note that there is a slim difference between extension and cause/effect relationship. Indeed, the extension relationship concerns the complementarity between rules without necessarily having an extend rule that activates a base rule. However the cause/effect relationship concerns the activation's rule without necessarily having a functioning complementarily between the cause and base rules. Another point is that, the included and extension relationships are manually defined by a designer, while cause/effect relationship can be detected automatically by analyzing the events and post event parts in rules.

In order to avoid anomalies in the definition of relationships between business rules (e.g., a cycle of relationships), the following properties must be verified:

	Asymmetric	Transitive	Reflexive	Euclidean
Inclusion (I)	Yes	Yes	No	No
Extension (E)	Yes	Yes	No	No
Cause/Effect (CE)	Yes	No	No	No

Firstly, an inclusion relationship is asymmetric because for all R_i and R_j in the Rules set, if R_i I R_j then not R_j I R_i since inclusion rule can not include the functionality of a base rule. For the same reason, extension and cause/effect relationships are asymmetric. Secondly, inclusion and extension relationships are transitive because for all R_i, $R_{j,}$ and R_y in the Rules set, if R_i I (or E) R_j and R_j I (or E) R_y then R_i I R_y. In contrast, a cause/effect relationship is not necessary transitive because if R_i activates R_j and R_j activates R_y then R_i does not necessarily activate Ry. Thirdly, all these relationships are not reflexive because one rule does not include (extend) its proper functionality. A rule that activates itself is treated as an error because it makes a process never end. Finally, all these relationships are not Euclidean because for all R_i, $R_{j,}$ and R_y in the Rules set, if R_i I (E or CE) R_j and R_i I (E or CE) R_y then it is not necessary that R_j I (E or CE) R_y. In addition, we can have some isolates rules, which will be activated by an external event and which does not activate any rules.

The fact of defining relationships between business rules allows determining which rules must be revised in case of change. Firstly, all base rules which have an inclusion relationship with a changed inclusion rule must be revised by a business process designer. In the previous example, if the enterprise decides not to deliver its products rule R_4 will be deleted from the process model. The suppression of an inclusion rule (R_4) will affect a base rule, which requires the completion of the inclusion rule's action. Due to this, human intervention is required to decide how we can change a base rule in order to keep the process coherence. Secondly, all base rules which have an extend relationship must be revised when an extend rule is changed. In the previous example, if we change rule R_2 (rule responsible for costumer identification), which represents an extension rule, then base rule R_{12} (rule responsible for discount calculation) must be revised. Finally, all effect rules which have a cause/effect relationship must be revised if the cause rule is changed in order to ensure the activation of these rules. For example the consequence of removing rule R_2 in our previous process is the inactivation of R_3, because R_2 is the cause of activating R_3. For this purpose, a designer must revise the effect rules if the cause rule is changed.

3.2 Development of graph of rules

To formalize the flexibility management of a process model, we propose to translate a business process into a graph of rules. Indeed, vertices of this graph represent the business rules, which constitute the business process, and arcs represent the relationships between the various rules. Three types of arcs are identified: includes arcs that correspond to inclusion relationship; extend arcs that correspond to extension relationship, and cause/effect arcs that correspond to cause/effect relationship between rules. A graph of rules is formally defined as follows:

Definition1: a graph of rules is a directed graph G_r (R, Y) with

- R is a set of vertices that represent business rules.
- Y is a set of arcs that represent three kinds of relationships.

(1) Y_i is a sub set of Y such that if y_i (r_i, r_j) then r_i is included in r_j.

(2) Y_e is a sub set of Y such that if y_e (r_i, r_j) then r_i extend r_j.

(3) Y_c is a sub set of Y such that if y_c (r_i, r_j) then r_i cause the activation of r_j.

The rule graph of our previous example is illustrated by figure 2. An inclusion arc is represented by a dashed arrow with a small diamond head on the side of the base rule. And, an extend arc is represented by a dashed arrow. Finally, a cause/effect arc is represented by a plain arrow. Note that two vertices can be linked by two arcs. For instance, R_4 is linked with R_6 by cause arc and extend arc (because R_4 cause the activation of R_6 and in the same time R_4 extend R_6).

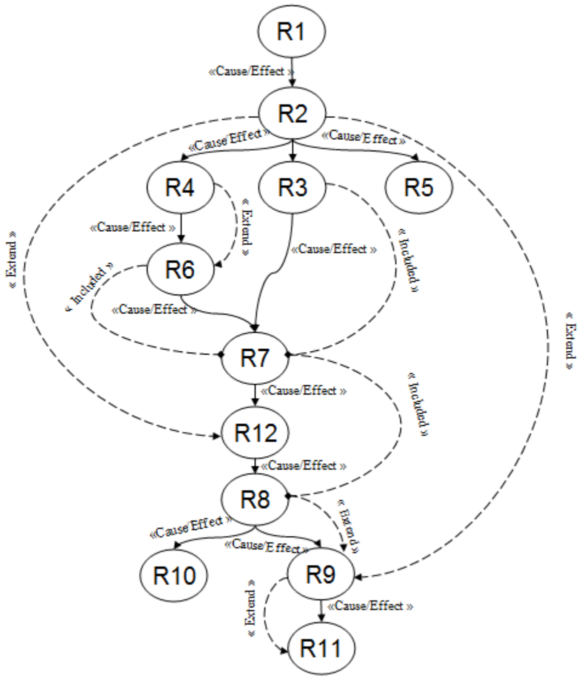


Fig.2. Rules graph of the purchase order process.

3.3 Change impact assessment

The graph of rules helps determine which rules are impacted by the change in a rule. Indeed, if any vertex changes, all **direct successor vertices** must be revised. Formally this will be defined as follows:

Definition 2: let G_r (R, Y) be a rule graph and r_i a vertex rule such that $r_i \in$ R.

The set of r_i direct successor neighbors is noted as $N^+(r_i)$ such that $\forall r_j \in N^+(r_i)$, r_i is either include, extend or, cause rule for the base or effect rule r_j.

- We note $N_i^+(r_i)$ the set of direct r_i successors such that $\forall r_j \in N^+(r_i)$, r_i is an include, for the base rule r_j.
- We note $N_e^+(r_i)$ the set of direct r_i successors such that $\forall r_j \in N^+(r_i)$, r_i is an extend rule for the base rule r_j.
- We note $N_c^+(r_i)$ the set of direct r_i successors such that $\forall r_j \in N^+(r_i)$, r_i is a cause rule for the effect rule r_j.
- We note $N_c^-(r_i)$ the set of direct r_i predecessors such that $\forall r_j \in N^-(r_i)$, r_j is a cause rule for the effect rule r_i.
- We note $N^*(r_i)$ the set of direct r_i neighbors such that $N^*(r_i) = N_i^+(r_i) \cup N_e^+(r_i) \cup N_c^+(r_i) \cup N_c^-(r_i)$. If $r_i \in R$ change, then the designer will have to revise all rules $N^*(r_i)$.

To keep a process coherent the flexibility management of the process modeling will request from a designer to revise the $N^*(r_i)$ set when a rule r_i is changed. In the example of figure 3, rule R_6 must be revised if rule R_4 is deleted because $N^*(R_4) = \{R_2, R_6\}$. The flexibility management notifies a business process designer to revise rule R_2 and R_6 in order to decide how this rule can be changed. Note that we must check out the direct predecessor neighbors $N_c^-(r_i)$ for the cause/effect relationship since it is not acceptable that a rule activates a non-existing rule. For instance, if R_4 is deleted we will also have to revise R_2 to ensure that this letter does not activate a deleted rule.

However, when changing the set of direct successor neighbor's include and extend rules ($N_e^+(r_i) \cup N_c^+(r_i)$) the designer should revise entirely the concerned rules. This revision may generate a cascade of rule change. Indeed, if one rule changes, the set of include and extend rules will be revised and properly changed. This will raise the need to revise another set of successor neighbor's rules of the rule that was revised. In the example, if R_4 is changed, then rule R_6 (extend rule) will be revised. This revision consists of analyzing the entire code of rule R_6 to decide how we can change the latter in order to keep the coherence of the process. If we change rule R_6 after its revision, this results in revising R_7. In turn, R_7 can be changed after revision, this results into revising R_8 and R_{12}. And so on, until we don't have any rule to revise.

In contrast, to change the set of direct successor neighbor's cause rules ($N_c^+(r_i) \cup N_c^-(r_i)$) that do not generate a cascade of the change because the designer, in this case, should only revise the event and post event part of the rules concerned. In the example, if we change R_4, then rules R_2 will be revised. This revision consists of updating the post event to ensure that this letter does not activate a deleted rule (as we explained above). After this update, we do not need to revise another set of direct successor neighbor's rules.

The following algorithm resumes the change impact of a rule

```
ChangeImpact_Procedure ($R_X$ , stack S)
{ if NotExist(S, $R_X$) then // test if the
rule's stack S contains the rule $R_X$
   { push (S, $R_X$); // push the rule $R_X$
onto stack S }
  if NotExist(S, $N_c^-$( $R_X$)) then $R_X$
     { push (S, $N_c^-$( $R_X$)); }
  if NotExist(S, $N_c^+$( $R_X$)) then
     { push (S, $N_c^+$( $R_X$)); }
  if $N_i^+$( $R_X$) $\neq \Phi$ then
   { ChangeImpact_Procedure ($N_i^+$( $R_X$),S);
   }Else
    { if $N_e^+$( $R_X$) $\neq \Phi$ then
      {ChangeImpact_Procedure ($N_e^+$($R_X$),S);
      }
      Else
       { exit ();}}
  }
```

It should be noted that a change cascade is not a consequence of the flexibility management that we propose. Indeed, flexibility management is not about implementing changes but about guaranteeing process consistency. In the previous process, rule R_4 change cascade (R_2, R_6, R_7, R_8, R_9, R_{10}, R_{11} and R_{12}) needs to be revised in order to ensure the activation of all the rules and the business coherence of the process as well. In the following we suggest how a designer is given the possibility of assessing the efforts to put into per change.

3.4 R2CM Model

In order to offer a tangible estimation of the efforts needed to implement rule changes, we propose a ***Rules Change Cost Model** (R2CM)*. Indeed, change cost is the necessary effort to modify the rules that are subject to changes following a change in a rule. For example If R_4 is deleted and R_6 is changed in the previous example, so the effective change effort applicable to R_4 concerns the effort to change R_4 plus the effort to change R_6. However, it is more beneficial to estimate the maximum change cost before making any changes. This will indicate to a designer the cost of a planned change. For this reason, in R2CM model the term "***cost of change***", denoted by $\zeta(R_i)$, is used to designate the maximum change cost before the change occurs.

The R2CM model is based upon a rule change impact graph which is derived from the graph of rules (figure 3). Indeed, the new graph is defined where vertices represent rules, arcs represent the relationships between the various rules, and there exists one vertex that represents the changed rule and does not have predecessors. Note that, the predecessor neighbors cause rules of the changed rule in the graph of rules become the successor neighbors cause rules in the change impact graph because these latter are impacted by the rule change (in the previous example, R_2 become a cause rule successor on the changed rule R_4).

By using a rule change impact graph, the R2CM model computes cost of change as a function of two parameters: (1) the distance between a changed rule and each impacted rule in this graph, and (2) the nature of the relationships between a changed rule and each affected rule in this graph.

Firstly, the overall cost of change of rule R_i is the sum of the cost change of the rules with different distances in a rule change impact graph. Formally this will be defined as follows:

$\zeta(\mathrm{R_i}) = \sum_{i=0}^{d_{\max}} C_i$ …. (1), where C_0 represents the cost of change of rule R_i. In the previous example $\zeta(R_4)$ = $C_0 + C_1 + C_2 + C_3 + C_4 + C_5$

However, according to Xiao et al. in [9], the nodes with shortest distance are more likely to be directly impacted by changes than the node with the longest distance away from the changed node. Consequently, the change cost of the rules with the shortest distance is greater than the change cost of the rules with the longest distance. Formally $C_i = \alpha C_{i-1}$ …(2), where α is a constant which is always between zero and one ($0 < \alpha < 1$).

In this way, from formulas (1) and (2) we deduct that the overall cost of change $\zeta(R_i)$ is a geometrical series with α as a constant ratio. The general term of this series is given as follows:

$$\zeta(\mathrm{R_i}) = C_0 \frac{1-\alpha^{d_{\max}}}{1-\alpha} \quad \text{…(3)}$$

Secondly, according to the nature of the relationships between rules, two qualifications for the change cost can be considered:

- We qualify **high** change cost (C_H) the effort to put into changing inclusion and extension relationships because the designer should revise entirely the rules concerned.

- We qualify **low** change cost (C_L) the effort to put into changing a cause/effect relationship because the designer should revise the event and post event part of the rules concerned.

In this way, the rules cost change with distance *i* denoted by C_i is defined as follows*:* $C_i = n_i c_{L_i} + m_i c_{H_i}$ …(4), where n_i is the number of the case rules at distance *i* and m_i is the number of include and extend rules at distance *i*. However, as explained above, the change cost of include and extend rules is higher then *change* cost of case rules. Consequently, formula (4) becomes: $C_i = (n_i \beta + m_i) c_{H_i}$ …(5), where β is a constant which is always between zero and one ($0 < \beta < 1$).

To sum up, *R2CM* model estimates the effort needed to implement the rule change by using the following formulas:

$$\begin{cases} \zeta(\mathrm{R_i}) = C_0 \dfrac{1-\alpha^{d_{\max}}}{1-\alpha}, \text{ such that } \alpha \in \left]0,1\right[\\ \forall i \in [1, d_{\max}], C_i = (n_i \beta + m_i) c_{H_i}, \text{ such that } \beta \in \left]0,1\right[\end{cases}$$

In order to demonstrate the result of the R2CM and to determine α and β values, we will apply, in the future, our approach on real BPMS by using series of experiments. For this reason, we will collect a set of business processes and express these processes using our rule based model. Afterward, we repeatedly apply our impact change approach on the changed processes in the aim of finding a set of consistent mathematical model of α and β (or interval). The consistency of this mathematical model that can help refine the values of α and β.

4 Related work

Our main motivations stem out of the importance of improving business process management in terms of flexibility. To this end we identified two major research directions: first, which rule formalism from the existing rule formalisms that allows managing the impact of changes? Second, how we can evaluate the impact of changes?

Firstly, the rule-based approach proposes to model the logic of the process with a set of business rules.

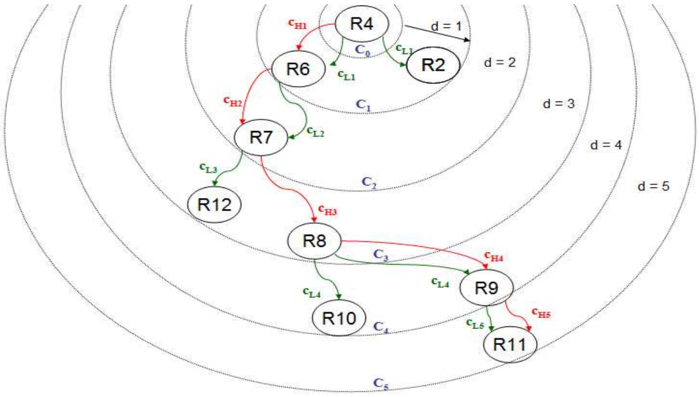


Fig.3. the change impact graph of the rule R_4

Indeed, according to Wagner in [10] the different structural categories of a business rule can be considered: (1) **Integrity rule**: it concerns the constraint or assertions that must be satisfied. (2) **Derivation rule**: it concerns one or more conditions and one or several conclusions. (3) **Production rule**: it concerns one or more conditions and one or more actions (4) **Reaction rule**: it concerns the rule that is triggered by events occurrences and which require the satisfaction of conditions to perform actions. (5) **Transformation rule**: it concerns the rule that control change of the system state. To formalize rule expressions different languages, are proposed. Indeed, the rules formalism used in these models depend to what categories of rule they represent. Note that, these languages can be used for interchanging rules between different rule languages like RuleML [11] and R2ML [10]. However, according to Knolmayer et al. in [12] the reaction rules (ECA) are the most adapted to model business rules. Giurca et al. in [8] justified this by the fact that this kind of rules is easier to maintain and it cover all other rules kinds (Integrity, deviation, production, and transformation).This is done in various works, like [12], [13] and [14]. Our work is positioned in ECA rule category. However, in the aforementioned declarative process modeling languages using this formalism, the modeling flexibility with focus on the impact of a rule change on the rest of a process is not well looked into. Therefore, there is a need for a more powerful formalism that would allow a complete definition of this relationship. This is why we proposed the ECAPE formalism.

Secondly, how to analyze the impact of software change is a research topic for several years. This is why literature teems with proposals that attempt to answer this delicate question. Like OMEGA project proposed in [15] which identify the propagation effects caused by code modification in C++ program. And also, the work of Lee et al. in [16] which proposes a new analysis technique for object-oriented software by using different dependency graphs (intra-method data dependency graph, inter-method data dependency graph, system dependency graph …etc) to calculate the impact propagation. Note that, some cost models are proposed in order to estimate necessary effort to a software development. An example of this is the famous COCOMO model [17]. In parallel, some works were interested in analyzing the impact of a business process changes. For instance, the work of Xiao et al. in [9] that proposes an approach to support impact analysis by using change impact metric. This metric is based on distance between a changed rule and each affected rules in a generated propagation graphs. This approach is, some where, similar to our proposed approach. However, the different between Xiao's approach and your, is that we proposed to model a business process as a set of rules. This allow, in the hand, deploying partially-specified process definitions. In other hand, definition of this relationship between rules in other to manage the impact of change efficacy.

5 Summary

In this paper we proposed a new rule based model that addresses the following issues: implementation of business rules in a business process code makes this process rigid and difficult to maintain, and flexibility of business modeling management. For this purpose, we suggested the ECAPE formalism to describe a business process using a set of business rules that are afterwards translated into a graph of rules. This graph is used to estimate the change cost by using a new cost change model called R2CM. In the future, we plan to demonstrate the result of our approach through empirical case studies by using an open source business application.

References

[1] S.Goedertier and J. Vanthienen, "Compliant and flexible business process with business rules." In 7th Workshop BPMDS'06 at CAiSE'06 pages: 94-104.

[2] The Business Rules Group, Defining Business Rules, What are they really? www.businessrulesgroup.org, July 2000.

[3] OASIS: Business Process Execution Language for Web Services (BPEL4WS): Version 2.0. In BPEL4WS specification report (2007).

[4] M. Zur Muehlen. M.Indulska and G.Kamp. "Business Process and Business Rule Modeling: A Representational Analysis". In: 3rd International Workshop on Vocabularies, Ontologies and Rules for the Enterprise (VORTE 2007), Annapolis, Maryland, USA.

[5] M.Boukhebouze, Y.Amghar, A.N.Benharkat and Z. Maamar. "Towards self-healing execution of business processes based on rules." In ICEIS:11th International Conference on Enterprise Information 2009, Springer ed. Milan, Italy.

[6] R.Lu and S.Sadiq. "A Survey of Comparative Business Process Modeling Approaches". In BIS 2007: 82-9429

[7] G.Regev, P.Soffer and R.Schmidt.R "Taxonomy of Flexibility in Business Processes". Seventh Workshop on Business Process Modeling, Development, and Support In conjunction with CAiSE'06.

[8] Giurca, A., Lukichev, S., Wagner, G., (2006) 'Modeling Web Services with URML.' In: Proceedings of SBPM2006, Budva, Montenegro (11th June 2006), June 2006

[9] H.Xia, J.Guo and Y.Zou. "Supporting Change Impact Analysis for Service Oriented Business Applications" in International Workshop on Systems Development in SOA Environments (SDSOA'07)

[10] G.Wagner. "Rule Modeling and Markup" in Reasoning Web, 3564 ed, N. Eisinger and J. Maluszynski, Eds. Msida, Malta: Springer, 2005, pp. 251-274.

[11] M.Schroeder and G.Wagner. "Languages for Business Rules on the Semantic Web". In Proc. of the Int. Workshop on Rule Markup Italy, June 2002.

[12] G.Knolmayer, R.Endl, and M.Pfahrer. "Modeling Processes and Workflows by Business Rules." In Business Process Management, Models, Techniques, and Empirical Studies, 2000.

[13] R.Müller, U.Greiner, and E.Rahm. "AgentWork: a Workflow System Supporting Rule-Based" Workflow Adaptation. In Data & Knowledge Engineering, 2004

[14] L.Zeng, A.Ngu, B.Benatallah, and M.O'Dell. "An Agent-Based Approach for Supporting Cross-Enterprise Workflows." In proceedings of the 12th Australasian Database Conference (ADC2001) (2001).

[15] X.Chen, W.Tsai, H.Huang, M.Poonawala, S.Rayadurgam, and Y.Wang, "Omega-an integrate environment for C++ program maintenance", In Software Maintenance Proceedings, CA, Nov. 1996, Page(s):114 – 123.

[16] M.L.Lee. "Change impact analysis of object-oriented software" in Master of Science, George Mason University, 1995.

[17] B.Boehm,. "Software Engineering Economics, Prentice Hall." ISBN: 0138221227, Prentice Hall (1981).