A Rule-Based Language for Integrating Business Processes and Business Rules

Tuan Anh Pham and Nhan Le Thanh

WIMMICS-INRIA Sophia Antipolis
2004 Route des Lucioles, 06902, Valbonne, France

{tuan-anh.pham, nhan.le-thanh}@inria.fr

Abstract. Business process modeling has become a popular method for improving organizational efficiency and quality. Automatic validation of process models is one of the most valuable features of modeling tools, in face of the increasing complexity of enterprise business processes and the richness of modeling languages. This paper proposes a formal language, Event-Condition-Action-Event (ECAE), for integrating Colored Petri Nets (CPN)-based business process with a set of business rules. We automate the integration process for validating the business process model. The ECAE language has several important features: its reasoning capabilities, its ability to express complex actions and events, and its declarative semantics. By enabling simulation of business process behavior, the reasoning capabilities facilitate the early detection of flaws

Keywords: Logic Programing; Business Process Management; Event-Condition-Action; Colored Petri Nets.

1 Introduction

The widespread use of business process modeling has helped enterprises to design, control and analyze many operational processes. Unfortunately, syntactic and semantic inconsistencies often appear in business process models, especially as the complexity of the models increases. Flaw detection and automation are essential for ensuring cost-effective and correct process models.

The challenge for system designers is to build a flexible intelligent system, which accepts and verifies the change on business process and business rules automatically. The business process must be integrated with a set of business rules, and a correspondence between the process and the rules must be created. This must be flexible since the business process and the business rules may be modified during runtime. The verification should be a rule-based system, which can reason and deduce new knowledge or a new decision based on a set of rules and facts.

This paper proposes a formal language ECAE for business process modeling, which takes advantage of both the graphical representation of colored Petri nets and the easy to represent ECA rule. It designs a business process model through CPN and translates

the model into a set of ECA rules, derivation rules and inhibition rules, it will be explained in more detail in section 5. This language can be used also for representing business rules and checking the respect of a business process to the business rules automatically when a user modifies a workflow.

Our main contributions in this paper are:

- Modeling a business process in a formal way.
- Representing a set of business rules in the same formal way with the business processes.
- Integrating the business process and a set of business rules.
- Checking the semantic aspect of business process automatically during runtime.

The rest of the paper is organized as follows. Section 2 presents a comparison with previous work. Section 3 introduces the research methodology. Section 4 provides an overview of both the Color Petri Nets and the ECA language. Section 5 presents the language ECAE through a case study. Finally, some conclusions and future research directions are presented in Section 6.

2 Comparison with related works

The most widely used languages for describing business process today are the Business Process Execution Language (BPEL) [16] and [BPMN]. BPEL and BPMN describe a process as a series of activities with a control flow (e.g., sequential execution) in an imperative fashion. Whereas traditional business process description languages center on activities, ECA rules put emphasis on events. In contrast, our approach specifies how to execute an action automatically when the event happens, provided that a certain condition holds. Another advantage of ECA rule-based approach is that it allows the users to specify requirements in either a natural or formal language, as business rules, legislative rules, or contractual rules. ECA rules easily integrate with other kinds of rules commonly used in business applications such as deductive rules (rules expressing views over data or rules used for reasoning with data) and normative rules (rules expressing conditions that data must fulfill; also called integrity constraints).

Several authors have proposed using ECA rules for business process modeling and execution, e.g., [10, 11, 12, 13, 14]. Some of these systems [8] use composite events to detect complex business process situations and only consider the structure and the execution of business process. By contrast, our ECAE language uses ECA rules for business process management: we address not only the structure and execution but also the problem of business constraints and integration with a set of business rules. To the best of our knowledge, no research work considers this aspect of ECA language. The only discussing transformation between CPN and ECA [17] does not consider CPN verification.

3 Research Methodology

Fig. 1 shows the iterative research process we apply. Initially, we conducted a requirements analysis based on case studies. This enabled us to formulate a state-of-the-art of business process validation. Based on the state of the art, we chose a theoretical basis, proposed a solution, and implemented it in prototypes. The solutions do not necessarily address all aspects of the requirements analysis at once but may rather focus on certain aspects. Using the prototypes we developed, we analyzed and evaluated our solutions using data from commercial applications. This may lead to a further iteration on the development and implementation (e.g., in case the developed concepts do not yet cover all relevant aspects or do not yet yield adequate solutions). The evaluation of the solutions developed may also result in a completely new iteration leading to modifications or refinements of the solution when studies reveal additional requirements.

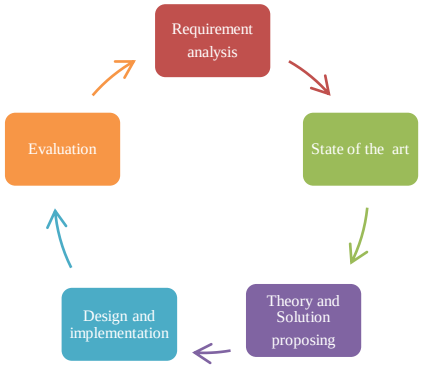

Fig. 1. Research Methodology

4 Background

4.1 Colored Petri Nets

In this paper, a Coloured Petri Net (CPN) is used to design a business process model. A CPN is a tuple CPN = (Σ, P, T, A, N, C, G, E, IN) [4], where Σ is a set of colors; P is a finite set of places; T is a finite set of transitions; A is a finite set of arcs; N is a node function; C is a color function; G is a guard function; E is an arc expression function; and IN is an initialization function.

The advantage of CPN is that color sets are used to distinguish different tokens, which will be treated in different ways, while arc expression function and guard function are used to control token's flow path.

Example 1: in Fig. 2, we design a CPN graph to represent bank account operations.

```
color Account = int with 1..1000;
color Balance = int;
color Amount = int with 1..5000;
color AB = product Account * Balance;
color AA = product Account * Amount;
var a:Account; var x:Amount;
var y:Balance;
```

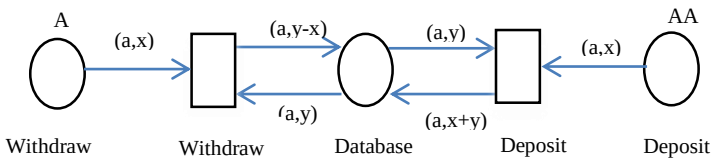

Fig. 2. Bank Account Operations

This provides a simple example. There are two main transitions in the CPN graph. The first transition allows the user to deposit the money to their bank account while the second action allows the user to withdraw the money from their bank account.

4.2 Event Condition Action

Event-condition-action (ECA) [17] rules are one way of implementing this kind of functionality. An ECA rule has the general syntax:

On event **If** condition **Do** actions (1)

The *event* specifies a condition for triggering the rule. The *condition* is a query, which determines if the information system is in a particular state, in which case the rule fires. Finally the *action* states the actions to be performed if the rule is met. These actions may in turn cause further events to occur, which may in turn cause more ECA rules to fire.

5 Sketch of the Proposed Solution

5.1 Overview of the Solution

In Fig 3, there are three main steps of our solution. First of all, a business process (CPN graph) is designed by a user; it contains all the properties of CPN (Places, Transitions, Input Arcs, Output Arcs, GuardFunctions, InputArc Expressions, OutputArcExpressions, Colour Sets). The business process can be modified and reused by the user. The second step is compilation; the business process which was designed in step 1, will be translated into a set of ECAE language rules, an extension of Event

Condition Action language. This language and the compilation step will be introduced in sections 4.2 and 4.3. Finally, in the execution step, the ECAE language will be executed with the ECA engine.

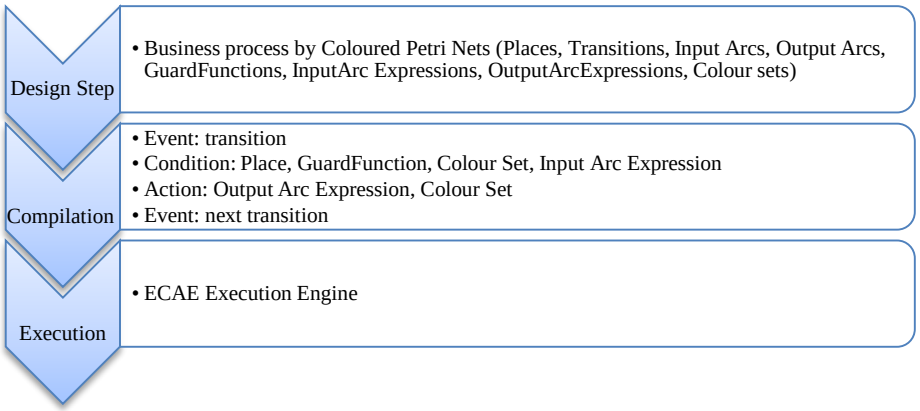

Fig. 3. Sketch of the solution

5.2 Outline of the Event-Condition-Action-Event (ECAE) Language

We start this section by informally introducing the various constructs of the language. Our solution is inspired by previous work [7]. In this solution, we aim at defining a language exhibiting both the advantages of ECA languages and of Logic Programing [18] updates. As such, expressions in ECAE are divided into two parts:

1. Rules: reactive rule, inference rule and inhibition rules.
2. Definitions: object, event and action

Reactive rules are as usual in ECA languages, and have the form (1), where: Event is a basic or a complex event expressed in algebra; Condition is a conjunction of (positive or negative) literals and Action is a basic or a complex action. Inference rules are Logic Programing rules with default negation, where default negated heads are allowed. Finally, ECAE also includes inhibition rules of the form:

If Condition **Do Not** Action

If Condition **Do** Action

Where *condition* is a conjunction of literals and events. Such an expression intuitively means: if *Condition* is true, do not execute *Action*. Inhibition rules are useful for updating the behavior of reactive rules. If the inhibition rule above is asserted all the rules with *Action* in the head are updated with the extra condition that *Condition* must not be satisfied in order to execute *Action.*

ECAE allows basic events to be combined to obtain complex ones using event algebra. The operators we use are: $\wedge \mid \vee \mid S \mid$ not. Intuitively, $e_1 \wedge e_2$ occurs at an instant i if both e_1 and e_2 occur at i; $e_1 \vee e_2$ occurs at instant i if either e_1 or e_2 occur at instant i;

not e occurs at instant i if e does not occur i. S (e_1, e_2, e_3) occurs at the same instant of e_3, in case e_1 occurred before, and e_2 in the middle. Operator S is very important since it allows combining and reasoning with events occurring at different time points.

Actions can be basic or complex. Basic external actions are related to the specific application of the language. Basic internal actions are for adding or retracting facts and rules (inference, reactive or inhibition rules), of the form assert(τ) and retract(τ) respectively, for raising basic events, of the form raise(e). There is also an internal action defined (d) for adding new definitions of actions and events (see more on these definitions below).

Complex actions are obtained by applying algebraic operators on basic actions. Such operators are: $\Rightarrow$| $\Updownarrow$ | IF, the first for executing actions sequentially, and the second for executing them concurrently. Executing IF (C, a_1, a_2) amounts to executing a_1 in case C is true, or executing a_2 otherwise.

To enable modular definition of both complex actions and events, ECAE allows for event and action definition expressions. These are of the form e_{def} **is** e and a_{def} **is** a where e_{def} (resp. a_{def}) is an atom representing a new event and e (resp. a) is an event (resp. an action) obtained by the event (resp. action) algebra above. It is also possible to use defined events (resp. actions) in the definition of other events (resp. actions).

5.3 Translation from CPN Business Process Model to ECAE Rule

As mentioned in section 4.1, a business process is represented by a CPN graph; the idea of our solution is to translate a CPN graph to a set of ECAE rules. We propose an algorithm for CPN-ECAE translation:

The ECAE rules, translated from Coloured Petri Net-based business process model is used to realize business process execution. The translation algorithm has 4 steps as follows:

1. The condition part of ECAE reactive rule is a collection of color sets, guard function related to a transition.
2. Translate each transition to ECAE rule
3. Add starting condition and ending condition.
4. Connect all ECAE rule transition as their triggered sequence.

With this algorithm, we can translate a business process model into a set of ECAE rules. Example 2 illustrates this algorithm.

Example 2: the CPN graph from Example 1 will be translated into a set of ECAE rules.

```
BPR1:If Withdraw&AA(a,x)Do Withdraw&AB(a,y-x)
BPR2:On Withdraw&AB(a,y-x) If Done Do AB(a,y)
BPR3:If Deposit&AA(a,x) Do Deposit&AB(a,y+x)
BPR4:On Deposit&AB(a,y+x) If Done Do AB(a,y)
BPR5:On AB(a,y) If Done Do EndWorkflow
BPR6:If Account>5000&Account<0 Do EndWorkflow
```

BPR$_7$:If Amount>1000&Amount<0 **Do** EndWorkflow

In this example, R_1 and R_3 are the rules to begin the business process for two cases, Withdraw and Deposit, respectively. R_5 is the rule for quitting the business process.

5.4 Business Rules

One of the main objectives of ECAE is to build a set of business rules. When a business process is executed, it must respect a set of business rules. A rule set consists two parts:

1. Definitions: this part contains all definitions of actions, events and color set in a specific domain.
2. Inhibition rules: this part consists a set of inhibition rules which are useful for updating the behavior of reactive rules

Example 3: we extend Example 2 by adding some actions and simple inhibition rules.

BRR$_1$: Withdraw(a,x) **is** Login(user,pass)$\Rightarrow$Amount(a,y-x)
BRR$_2$: Deposit(a,x) **is** Login(user,pass)$\Rightarrow$ Amount(a,y+x)
BRR$_3$:If y-x<0 **Do No**t Withdraw(a,x)
BRR$_4$:If Not Login(user,pass) **Do** EndWorkflow

When these inhibition rules are integrated with the set of ECAE rules in Example 2, the balance of bank account will never be negative. We can use ECAE to define this more complex business rule.

5.5 Verifying the Compliance of a Business Process with Business Rules

This section introduces our method for integrating and verifying a business process and business rules. As presented above, the set of business rules and business processes are represented by ECAE language. Therefore, in order to verify the compliance between them, we merge two sets of ECAE rules into a single knowledge base and reason on it. Let us continue our Example 3 we have a knowledge base as follow:

BPR$_1$:If Withdraw&AA(a,x)**Do** Withdraw&AB(a,y-x)
BPR$_2$:On Withdraw&AB(a,y-x) **If** Done **Do** AB(a,y)
BPR$_3$:If Deposit&AA(a,x) **Do** Deposit&AB(a,y+x)
BPR$_4$:On Deposit&AB(a,y+x) **If** Done **Do** AB(a,y)
BPR$_5$:On AB(a,y) **If** Done **Do** EndWorkflow
BPR$_6$:If Account>5000&Account<0 **Do** EndWorkflow
BPR$_7$:If Amount>1000&Amount<0 **Do** EndWorkflow
BRR$_1$: Withdraw(a,x) **is** Login(user,pass)$\Rightarrow$Amount(a,y-x)
BRR$_2$: Deposit(a,x) **is** Login(user,pass)$\Rightarrow$ Amount(a,y+x)
BRR$_3$: If y-x<0 **Do No**t Withdraw(a,x)
BRR$_4$:If Not Login(user,pass) **Do** EndWorkflow

We can see that the business process and the business rules are represented in ECAE syntax (this is a set of rules). Therefore, we can easily check the compliance of business process with a set of business rules by detecting the conflict between the rules in one knowledge base using reasoning and a reasoner.

6 Discussion and Conclusions

CPNs and ECA rules have a very important role in designing a business process management system. Colored Petri nets, inherited from the traditional Petri nets, have a better ability on expressiveness because of their color sets and guard function. Meanwhile ECA rules are based on the event-trigger feature, which is an easy-to-implement software initiative.

In this paper, we propose a formal language ECAE, which exhibits the advantages of both ECA languages and Logic Programing updates. Further, we design a common business process model for bank account operations using a colored Petri net, and translate it into a set of ECAE rules

In future work, we will focus on enhancing the expressiveness and exception processing ability of ECA rules, which will make our method more suitable for developing a useful business process management system. We will also consider the transaction problem for business process execution, and how to implement and evaluate the proposed approach based on process agents and ECA rules.

References

1. Ryan K.L. Ko, Stephen S.G. Lee, and Eng Wah Lee, "Business process management (BPM) standards: a survey," Business process Management Journal, vol. 15, pp. 744--791, 2009.
2. Marc Fasbinder, Why model business processes?, 2007.
3. L. J. Hommes, "The Evaluation of Business process Modeling Techniques," Delft University of Technology, Ph.D. thesis 90-9017698-5, 2004.
4. Liu Feng, Zhang Wei. Colored Petri net extended with price information and its applicaion[J]. Journal of Computer Applications; 2007, 20(10):2501-2503.
5. Nguyen, T.H.H., Le-Thanh, N.: An ontology-enabled approach for modelling business process es. In: Beyond Databases, Architectures, and Structures. Volume 424 of Communications in Computer and Information Science. Springer International Publishing (2014) 139-147.
6. Tuan Anh Pham, Thi-Hoa-Hue Nguyen, Nhan Le Thanh:Ontology-based business process validation. RIVF 2015: 41-46.
7. José Júlio Alferes, Federico Banti, Antonio Brogi: An Event-Condition-Action Logic Programming Language. JELIA 2006: 29-42.
8. Donghui Lin, Huanye Sheng, Toru Ishida: Interorganizational Business process Execution Based on Process Agents and ECA Rules. IEICE Transactions 90-D (9) (2007): 1335-1342.
9. George Papamarkos, Alexandra Poulovassilis, and Peter T. Wood: Event-condition-action rules on RDF metadata in P2P environments. Computer Networks 50(10): 1513-1532 (2006).
10. [10] D. Barbará, S. Mehrota, M. Rusinkiewicz. INCAS: A Computation Model for Dynamic Workflows in Autonomous Distributed Environments. Technical Report, Department of Computer Science, University of Houston, May 1994.
11. C. Bussler, S. Jablonski. Implementing Agent Coordination for Business process Management Systems Using Active Database Systems. Proc. 4 th RIDE-ADS, Houston, February 1994.
12. Joonsoo Bae, Hyerim Bae, Suk-Ho Kang, Yeongho Kim: Automatic Control of Business process Processes Using ECA Rules. IEEE Trans. Knowl. Data Eng. 16(8): 1010-1023 (2004)
13. Geppert, A., Tombros, D.: Event-based distributed business process execution with EVE. In: Proc. of the IFIP Int. Conf. on Distributed Systems Platforms and Open Distributed Processing, pp. 427–442 (1998).
14. George Papamarkos , Ra Poulovassilis , Peter T. Wood : RDFTL : An Event-Condition-Action Language for RDF. In Proc. 3rd Int. Workshop on Web Dynamics (in conjunction with WWW (2004), pp.223-248.
15. Alexandra Poulovassilis, George Papamarkos, Peter T. Wood: Event-Condition-Action Rule Languages for the Semantic Web. EDBT Workshops 2006: 855-864
16. Andrews, T., et al.: Business process execution language for web ervices version 1.1. Available at www.ibm.com/developerworks/library/ws-bpel (2003)
17. ZHOU, Guo-xiang et GAO, De-ping. ECA rule and colored Petri nets based workflow modeling research. The National Natural Science Foundation of China, 2010, p. 1-4
18. Sandro Etalle, Miroslaw Truszczynski: Logic Programming, 22nd International Conference, ICLP 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings. Lecture Notes in Computer Science 4079, Springer 2006, ISBN 3-540-36635-0